UNISYS

OS 2200
Interactive Processing Facility (IPF 1100) Command Language
User's Guide

Copyright © 1997 Unisys Corporation.
All rights reserved.
Unisys is a registered trademark of Unisys Corporation.

Release SB7 September 1997

Printed in U S America
Priced Item 7833 3788-002

NO WARRANTIES OF ANY NATURE ARE EXTENDED BY THE DOCUMENT. Any product and related material disclosed herein are only furnished pursuant and subject to the terms and conditions of a duly executed agreement to purchase or lease equipment or to license software. The only warranties made by Unisys, if any, with respect to the products described in this document are set forth in such an agreement. Unisys cannot accept any financial or other responsibility that may be the result of your use of the information in this document or software material, including direct, indirect, special, or consequential damages.

You should be very careful to ensure that the use of this information and/or software material complies with the laws, rules, and regulations of the jurisdictions with respect to which it is used.

The information contained herein is subject to change without notice. Revisions may be issued to advise of such changes and/or additions.

RESTRICTED RIGHTS LEGEND—Use, reproduction, or disclosure is subject to the restrictions set forth in DFARS 252.227-7013 & 252.211-7015/FAR 52.227-14 and 52.227-19 for commercial computer software, as applicable.

Correspondence regarding this publication should be forwarded using the Business Reply Mail form in this document, or remarks can be addressed directly to Unisys Corporation, PI Response Card, P.O. Box 64942, St. Paul, Minnesota, 55164-9544, U.S.A. Remarks can also be sent to the following internet e-mail address: rsvldoc@rsvl.unisys.com

TRADEMARKS—All terms mentioned in this document that are known to be trademarks or service marks have been appropriately capitalizes. Unisys Corporation cannot attest to the accuracy of this information. Use of a term in this document should not be regarded as affecting the validity of any trademark of service mark.

Contents

About This Guide xiii

Section 1. Introduction

1.1. IPF 1100 Overview 1-1
1.2. Components of IPF 1100 1-1
1.3. Command Language Overview 1-3
1.4. Scrolling the Display 1-3
1.5. Learning about the Keyboard 1-3
1.5.1. Cursor Positioning and the XMIT Key 1-3
1.5.2. The RETURN Key 1-4
1.5.3. The ? Key and UA 1100 1-4
1.6. Receiving Feedback on What You Enter 1-5
1.7. Controlling Screen Output Paging 1-6
1.7.1. $PAGING, $PAGELENGTH, and PAGEWIDTH 1-7
1.7.2. $WAITTIME 1-8
1.7.3. $PAGECALL 1-8
1.8. Using IPF 1100 Commands 1-8
1.8.1. Command Format 1-9
1.8.2. Keyword Parameter Format 1-10
1.8.3. Command and Keyword Abbreviations 1-11
1.9. Format Conventions 1-12
1.9.1. Comments 1-12
1.9.2. Continuation Lines 1-12
1.9.3. Multiple Commands on One Line 1-12
1.9.4. Identifiers 1-13
1.9.5. Labels 1-13
1.9.6. Constants 1-13
1.9.7. Conditions 1-13
1.9.8. Strings 1-14
1.9.9. Expressions 1-14
1.9.10. Unary and Binary Operators 1-16
1.9.11. Variables 1-17
1.9.12. Lists 1-17
1.10. Using Positional Notation 1-18

1.11. Calling and Leaving IPF 1100 1-19
1.12. Signing Off Your System 1-20

Section 2. Directories and Files

2.1. Working with Directories and Files 2-1
2.2. Naming Files 2-1
2.2.1. Using Read and Write Keys 2-2
2.2.2. Using Cycles 2-3
2.2.3. Using Version Names 2-3
2.3. Using Data Files 2-3
2.4. Assigning a Home Directory 2-4
2.5. Defining Work Directories 2-5
2.6. Data Images and Screen Lines 2-6
2.7. Creating Files and Directories 2-6
2.8. Creating Public and Private Files 2-7
2.9. Creating Temporary Files 2-8
2.10. Listing Files in a Directory 2-9
2.11. Deleting a File 2-10
2.12. Copying a File 2-11
2.13. Printing a File 2-13
2.14. Maintaining Files 2-15
2.14.1. PACK Command 2-15
2.14.2. ERASE Command 2-16
2.15. Specifying an Additional Name for a File 2-16
2.16. Explicitly Assigning a File 2-17
2.17. Explicitly Releasing a File 2-19
2.18. Retrieving Information about Files 2-20
2.19. Listing Files Associated with a Session 2-21
2.20. Working with Tapes 2-22
2.20.1. Mounting a Tape 2-23
2.20.2. Moving a Tape 2-23
2.20.3. Copying Data to and from Tapes 2-24
2.21. Cataloging Tape Files 2-25
2.22. Releasing a Tape File 2-25

Section 3. The Workspace and the Lookspace

3.1. Using the Workspace and the Lookspace 3-1
3.2. The Workspace 3-2
3.3. Creating a New File in the Workspace 3-4
3.4. Saving a New File from the Workspace 3-6

3.5. Retrieving an Existing File 3-8
3.6. Saving an Old File from the Workspace 3-9
3.7. Changing the Name of the Workspace 3-11
3.8. Changing the Subtype of the Workspace 3-12
3.9. The Lookspace 3-14

Section 4. Running Programs

4.1. Introduction 4-1
4.2. Running User Programs 4-2
4.3. Calling a Procedure or System Processor 4-6
4.4. Specifying the Search Order 4-8
4.5. Specifying an Absolute 4-10
4.6. Executing Commands 4-12
4.7. Interprogram Communication 4-12
4.7.1. The Run Condition Word 4-12
4.7.2. The CONDITION Command 4-13

Section 5. Variables and Functions

5.1. Introduction 5-1
5.2. Using System Variables 5-1
5.3. Giving a Value to a Variable 5-7
5.4. Printing the Values of Variables 5-7
5.5. Listing Variable Names 5-8
5.6. Soliciting a User Variable Value 5-9
5.7. Specifying System Functions 5-9
5.7.1. Working with Strings 5-10
Finding the Length of a String ($LENGTH) 5-10
Finding the Position of a Substring in a String ($SEARCH) 5-10
Returning a Substring within a String ($SUBSTRING) 5-11
Converting a String to All Lowercase Characters ($LOWERCASE) 5-11
Converting a String to All Uppercase Characters ($UPPERCASE) 5-12
Removing Blanks from a String ($TRIM) 5-12
Adding a "pad" to a String ($PAD) 5-13
Finding What Is At a Specific Line Number ($TEXT) 5-14
5.7.2. Working with Numbers 5-14

Finding the Absolute Value of a Number ($ABS) 5-14
Finding the Integer Portion of a Number ($INTEGER) 5-15
Finding the Fractional Portion of a Number ($FRACTION) 5-15
5.7.3. Converting Values from One Type to Another 5-15
Converting a Numeric Field to a String ($STRING) 5-15
Changing a String Expression to a Number ($NUMBER) 5-16
Finding the Integer Representation of an ASCII Character ($ASCII) 5-16
Finding the ASCII Character of an Integer ($CHARACTER) 5-16
5.7.4. Miscellaneous Functions 5-17
Retrieving Information from the Run Condition Word ($CONDITION) 5-17
Finding the Fully Qualified Filename ($FILE) 5-18
Getting Information on a Specific Field of a Form ($FIELDSTATUS) 5-19
5.7.5. Using More Complex Expressions 5-19

Section 6. Redirecting Input and Output

6.1. Standard I/O Paths 6-1
6.2. Providing Input from a File 6-1
6.3. Marking the End of Input 6-2
6.4. Sending Output to a File 6-5
6.5. Sending Error Messages to a File 6-6

Section 7. Command Reference

7.1. Format Conventions 7-1
7.2. Command Formats 7-1
7.2.1. ACCEPT Command 7-1
7.2.2. ATTACH Command 7-3
7.2.3. "Call" Command 7-5
7.2.4. CONDITION Command 7-7
7.2.5. COPY Command 7-8
7.2.6. CREATE Command 7-12
7.2.7. DESCRIBE Command 7-18
7.2.8. DISPLAY Command 7-19

7.2.9.	*END Command	7-20
7.2.10.	END_INPUT Command	7-20
7.2.11.	ERASE Command	7-21
7.2.12.	ERROR Command	7-22
7.2.13.	EXECUTE Command	7-23
7.2.14.	FREE Command	7-24
7.2.15.	IN Command	7-26
7.2.16.	INVENTORY Command	7-27
7.2.17.	LIBRARY Command	7-27
7.2.18.	LIST Command	7-28
7.2.19.	LOGOFF Command	7-30
7.2.20.	MOUNT Command	7-31
7.2.21.	NAME Command	7-35
7.2.22.	NEW Command	7-36
7.2.23.	OLD Command	7-37
7.2.24.	OUT Command	7-39
7.2.25.	PACK Command	7-40
7.2.26.	POSITION Command	7-41
7.2.27.	PURGE Command	7-43
7.2.28.	REPLACE Command	7-44
7.2.29.	REVIEW Command	7-45
7.2.30.	RUN Command	7-46
7.2.31.	SAVE Command	7-48
7.2.32.	SET Command	7-49
7.2.33.	SWITCH Command	7-50
7.2.34.	TYPE Command	7-51
7.2.35.	USE Command	7-52
7.2.36.	XQT Command	7-53

Bibliography

Index

Figures

1-1. IPF 1100 Components and Related OS 2200 Products 1-2

Tables

5-1. IPF 1100 System Variables for Command Language 5-2
5-2. IPF 1100 System Functions 5-19

About This Guide

Purpose

The OS 2200 Interactive Processing Facility (IPF 1100) is an easy-to-use timesharing and batch processing system used to update databases, develop and run programs, and call system processors. Command Language is one of several IPF 1100 components and is most commonly used with the EDIT 1100 and Procedures components.

This guide explains the IPF 1100 Command Language commands you use to create, modify, and manipulate files and run programs.

Scope

This guide contains introductory information about IPF 1100 and your system. It:

- Describes files and directories
- Describes the workspace and the lookspace
- Explains how to run programs
- Provides basic and detailed information about IPF 1100 commands that pertain to Command Language.

Audience

This guide is intended for anyone who uses IPF 1100 to work with files or to run programs.

Prerequisites

None.

How to Use This Document

If you are unfamiliar with IPF 1100, begin with Section 1. Sections 2, 3, 5, and 6 explain the commands, system variables, and system functions you use when working with files. If you are an experienced user, you may find the command reference in Section 7 the most useful. If you are running programs refer to Section 4.

Organization

This guide contains seven sections.

Section 1. Introduction

This section describes IPF 1100 and how Command Language relates to the other components. It also provides introductory information about your terminal and describes the IPF 1100 command format.

This section also shows how to start and end your IPF 1100 session.

Section 2. Working with Directories and Files

This section explains directories and files and how to use them in IPF 1100. It describes the basic commands you use to create and maintain files and directories.

Section 3. Using the Workspace and the Lookspace

This section explains how to use the IPF 1100 workspace to edit files as well as how to use the lookspace to browse through files.

Section 4. Running Programs

This section shows how to run user programs, call system processors, call IPF 1100 procedures, and execute commands.

Section 5. Using Variables and Functions

This section explains system and user variables and system functions. It shows how to assign values to the variables and how to display them.

Section 6. Redirecting Input and Output

This section explains how to have IPF 1100 read your input from a file. It also shows how IPF 1100 can send to a file the output it normally displays on the screen.

Section 7. Command Reference

This section provides more detailed information about the commands presented in Sections 2 through 6, including keyword parameters and positional notation rules.

Related Product Information

The following documents may be helpful to you when you work with IPF 1100. Use the version that corresponds to the level of software in use at your site.

Directly Related Documents

***OS 2200 Interactive Processing Facility (IPF 1100) Administration Guide* (7833 3754)**

This guide provides information to help you manage and maintain IPF 1100.

***OS 1100 Interactive Processing Facility (IPF 1100) EDIT 1100 User's Guide* (7833 3812)**

This guide explains the commands you use to update data in your workspace, emphasizing the full-screen mode commands. It also shows how to specify line ranges and use the pattern-matching feature.

***OS 1100 Interactive Processing Facility (IPF 1100) User Assistance (UA 1100) User's Guide* (7833 3762)**

This guide explains how to get online help when using IPF 1100. It also shows how to use the UA 1100 online tutorials.

***OS 2200 Interactive Processing Facility (IPF 1100) Procedures User's Guide* (7833 3770)**

This guide explains how to write procedures to make the IPF 1100 Command Language more powerful. Procedures allow you to save groups of commands together, including commands to change the flow of control when calling the procedure.

***OS 2200 Interactive Processing Facility (IPF 1100) User's Quick Reference Guide* (7833 3804)**

This quick-reference guide provides an alphabetized listing all the IPF 1100 commands, system variables, and system functions. It also provides other reference information.

***OS 2200 Interactive Processing Facility (IPF 1100) User's Reference Handbook* (7833 3820)**

This pocket-sized foldout card summarizes all the IPF 1100 commands, system variables, system functions, and other information.

***OS 2200 Interactive Processing Facility (IPF 1100) Error Message Manual* (7833 4018)**

This manual provides a listing of all IPF 1100 error messages and their online explanations.

***OS 2200 Interactive Processing Facility (IPF 1100) Master Index* (7833 3747)**

This master index provides a comprehensive index to the guides in the IPF 1100 library.

Other Related Documents

***OS 1100 UDS Relational Database Management System (UDS RDMS 1100) and IPF SQL Interface End Use Guide* (7831 0778)**

***OS 2200 UDS Relational Database Management System (UDS RDMS 2200) Programming Reference Card* (7831 0943)**

***OS 2200 UDS Relational Database Management System (UDS RDMS 2200) SQL Programming Reference Manual* (7830 8160)**

***OS 2200 Distributed Data Processing File and Job Transfer (DDP-FJT) Implementation and Administration Guide* (3787 3544)**

***OS 2200 Distributed Data Processing File and Job Transfer (DDP-FJT) IPF and Batch Interface Operations Guide* (3787 3551)**

***OS 2200 Meta-Assembler (MASM) Programming Reference Manual*(7830 8269)**

Notation Conventions

This guide uses the following conventions in command formats and examples:

COPY	Uppercase words represent command names, keyword parameters, system variables and system functions. You must type keywords exactly as they appear.
file	Lowercase italic words represent user-supplied values. User-supplied values appear for keyword parameters, system variables, and system functions and user input. Integers are appended to user-supplied values when they appear more than once in a format.
[	Optional words and phrases appear in brackets. For example, [*parameter-1*
{ }	Portions of commands where a choice of one (and only one) of a set of words, phrases, or values appear in braces. For example {*parameter-1* | *parameter-2* | *parameter-3*}
|	A vertical line separates the choices.

()	Portions of a command where you can choose one or more of a set of words, phrases, or values appear in parentheses. A vertical line separates the choices.
...	Portions of a command that may be repeated any number of times are identified by an ellipsis. An ellipsis occurs immediately following a right bracket or right brace and indicates that everything between that bracket or brace and its pared left bracket or brace may be repeated any number of times.
::	Use two colons as a separator within a *user-supplied value*. Use it whenever the *user-supplied-value* needs a *host-id* qualifier. The *host-id* qualifier is needed for various user-supplied values when using DDP 1100. (See the *DDP File and Job Transfer IPF and Batch Interface Operations Guide*.) For example, CREATE FILE=[*host-id*::] *filename*.
XMIT	Represents pressing the transmit key.
①	Lines on a screen that require explanation have numbers inside circled numbers. The corresponding explanations immediately follow the screen.
BOLD	Uppercase bold words in command and keyword formats represent the default value. Default values are indicated when more than one option is available for a command or keyword.

See Section 1 for command and keyword formats and conventions for items such as comments, continuation lines, expressions, and operators.

Reader Response

If you have any questions or comments about this document, you can either mail or fax the Business Reply Mail Form at the back of this document, or send comments to the following Internet E-mail address:

```
rsvldoc@rsvl.unisys.com
```

You can help us by including the following information:

- Your name
- Your company's name (optional)
- Your Internet E-mail address
- The document title and the 11–digit part number for the document (if applicable)

- The SB level or software level (software documents only)
- The system number and plateau level (hardware documents only)
- Any additional information that will help us in responding to your comments

Section 1
Introduction

This section introduces the Interactive Processing Facility (IPF 1100). It explains the conventions used in this manual, as well as some of the keys on your terminal keyboard that you will use often.

1.1. IPF 1100 Overview

Interactive Processing Facility (IPF 1100) is an interactive and batch processing system that allows users to develop application programs, as well as use the processors and databases already available on the system. IPF 1100 provides an interface to the OS 2200 system through an easy-to-learn, keyword-oriented command language. It is designed to allow access to the computer system by all types of users, including novice end users and experienced programmers.

The main features of IPF 1100 are

- An easy-to learn, easy-to-remember command and response language
- The ability to create command procedures using flow-of-control statements, argument passing, and canned run streams
- Full-screen and line-by-line modes of operation
- Completion notices to reassure users that IPF 1100 has processed their request
- Online assistance with commands or error messages at any time
- An SQL interface to Relational Data Management System (RDMS 2200) databases
- Direct access to the entire distributed data processing network

1.2. Components of IPF 1100

IPF 1100 consists of the following components:

- Control—Provides the environment and control structures necessary to use the other IPF 1100 components.
- Command Language—Provides the basic command services that let IPF 1100 users communicate with the operating system programs, work with files, manipulate user-defined variables, and redirect terminal input and output.
- EDIT 1100—Provides both a line mode and full-screen mode text editor.

- Procedures—Provides additional command language flexibility by allowing IPF 1100 users to define and store a group of IPF 1100 commands so they can be executed as one command.
- User Assistance (UA 1100)—Provides online assistance and explanations of error messages, commands, system variables and functions. UA 1100 also includes tutorials about IPF 1100.
- IPF Structured Query Language (IPF SQL)—Provides a SQL interactive ad hoc query interface to relational databases via the IPF 1100 environment. IPF SQL can display and update relational databases maintained by the Relational Data Management System (RDMS 2200).

IPF 1100 provides distributed processing capabilities through the following OS 2200 products:

- Distributed Data Processing File and Job Transfer (DDP-FJT) DDP-FJT controls the movement of files and jobs in a distributed network. By entering IPF 1100 commands, users can create, copy, and delete files on other hosts in the network, as well as submit jobs for processing.
- TCP/IP Application Services (TAS) TAS allows OS 2200 hosts to participate in networks that use the Transmission Control Protocol/Internet Protocol (TCP/IP) for interhost communications. For IPF 1100 users, TAS provides mail processing, using the simple mail transfer protocol (SMTP), and file transfer, using the file transfer protocol (FTP).

  Note: *In previous releases, TAS was called Defense Data Network (DDN 1100).*

  Figure Figure 1-1 shows the IPF 1100 components and the OS 2200 products IPF 1100 uses for distributed processing.

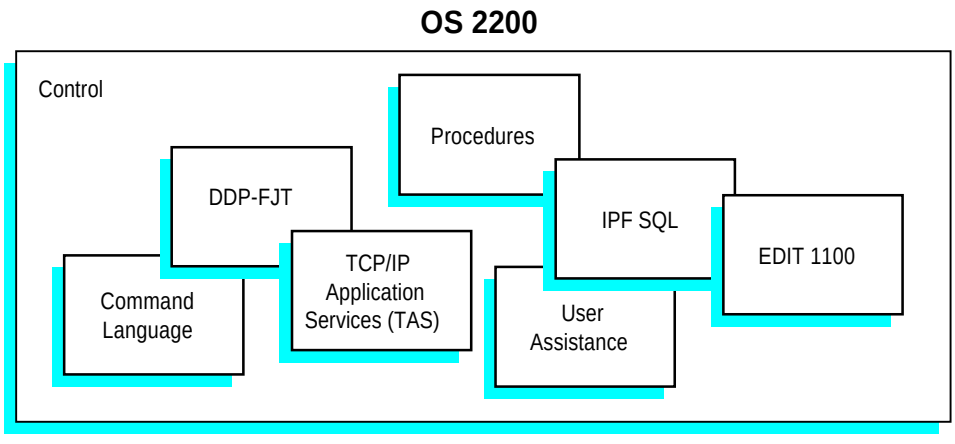


Figure 1-1. IPF 1100 Components and Related OS 2200 Products

1.3. Command Language Overview

Command Language provides commands, variables, and functions you use to manipulate files and directories. Command Language has

- A consistent format, with easy-to-learn and easy-to-remember keywords
- A set of consistent abbreviations for commands and keywords
- Clear, concise, easy-to-understand messages and command responses (known as completion notices)

To change the data in your files, you first make a copy of the file in your workspace. You use EDIT 1100 to edit the workspace, then copy the updated workspace back to your file.

1.4. Scrolling the Display

Whenever IPF 1100 displays a line of output on your screen, it appears as the bottom line of your screen. At the same time, all other lines on your screen move up one line, and the top line of your screen disappears. This is known as scrolling.

See 1.7 for information on controlling the "paging" of output to stop the information from scrolling off your screen.

1.5. Learning about the Keyboard

Here are some of the keys you will use often.

1.5.1. Cursor Positioning and the XMIT Key

The cursor (▒) marks your current position on the screen (on a UNISCOPE 100, you will actually see a . As you type, the cursor moves forward (toward the right side of your screen) one space at a time. You can use the cursor positioning keys on your terminal keyboard (←, →, ↑, ↓) to quickly move the cursor up, down, left, or right.

To send a screen, a partial screen, or a line of information to IPF 1100, press the XMIT (or TRANSMIT) key.

The position of the cursor on your screen when you press the XMIT key is very important. Only those lines and characters from the cursor back to the top of the screen or the first encountered start of entry (hereafter known as the SOE and seen on your terminal as ▶) are sent to IPF 1100. For example:

```
▸This is entered ▒  but this is not
```

Only "This is entered" is transmitted to IPF 1100. Unless you are in EDIT 1100 full-screen mode, you must press XMIT after each line you type. (In full-screen mode, you can transmit an entire screen of up to 20 lines at one time.)

1.5.2. The RETURN Key

Be aware that pressing the RETURN key on your terminal only sends your cursor to the first column of the next line on your screen. It does not send anything you have typed to IPF 1100. The only time you want to use the RETURN key instead of the XMIT key after typing lines is when using EDIT 1100 full-screen mode.

1.5.3. The ? Key and UA 1100

Anytime you need help while using IPF 1100, you can call User Assistance (UA 1100) by entering a question mark. (Press the ? key and then XMIT.) What you type before the question mark determines what type of information UA 1100 responds with.

Information is available about

- Messages IPF 1100 displays on your screen
- IPF 1100 commands and their formats

For more information on User Assistance, see the *User Assistance (UA 1100) User's Guide*.

1.6. Receiving Feedback on What You Enter

If you want to see messages telling you that a command (or operation) worked, you can tell IPF 1100 to display completion notices for most operations (e.g., after you have copied or deleted a file). Type SET $COMPLETIONS:=TRUE and press XMIT. $COMPLETIONS is a system variable that controls whether or not IPF 1100 displays these completion notices. See Section 6 for more information about system variables.

The default value of $COMPLETIONS is FALSE. Most examples in this manual assume a value of FALSE for $COMPLETIONS. The following example shows this distinction:

```
▸ ~C~▸SET $COMPLETIONS:=TRUE       ①
▸END SET       ②
▸ ~C~▸PURGE FILE=KLZ*BLUE.       ③
▸END PURGE       ④
▸ ~C~▸SET $COMPLETIONS:=FALSE       ⑤
▸ ~C~▸PURGE FILE=KLZ*RED.       ⑥
▸ ~C~▸▒       ⑦
```

Explanation

① Set the system variable $COMPLETIONS to TRUE.

② IPF 1100 displays a completion notice for doing this.

③ Enter a PURGE command to delete file KLZ*BLUE.

④ A completion notice verifies deletion of the file.

⑤ Change $COMPLETIONS to FALSE.

⑥ Enter a PURGE command to delete file KLZ*RED.

⑦ Since you requested no completion notices, IPF 1100 deletes the file and only returns the prompt and the cursor.

The full format of a completion notice is

```
END command-name x ERRORS y WARNINGS   z REMARKS
```

If your command results in any errors, warnings, or remarks, IPF 1100 includes the number of such messages in the completion notice. It will not display "x ERRORS", "y WARNINGS," or "z REMARKS" if their respective values are zero.

1.7. Controlling Screen Output Paging

Output paging is available in line mode and full-screen mode. The system variables $PAGING, $PAGELENGTH, PAGEWIDTH, $WAITTIME, and $PAGECALL control paging. Depending on the values of these system variables, IPF 1100 takes different actions when displaying output.

IPF 1100 uses the following rules to control page breaks:

- IPF 1100 always stops before it displays $PAGELENGTH+1 lines except during output from other processors.
- The following rules apply in full-screen mode:
  - A page break always occurs after a DESCRIBE or INVENTORY command unless the command is in a procedure or a multiple command line.
  - IPF 1100 always keeps the messages it generates intact. IPF 1100 issues a page break before a message rather than in the middle of the message if the message is long enough to cause a page break.
- $PAGECALL forces page breaks for the following commands:
  - "Call"
  - EXECUTE LANGUAGE=ECL
  - Tutorial invocations
  - XQT

1.7.1. $PAGING, $PAGELENGTH, and PAGEWIDTH

In line mode, $PAGING determines whether IPF 1100 generates page breaks. IPF 1100 automatically sets $PAGING to TRUE when you are in full-screen mode. If $PAGING is FALSE, IPF 1100 sends all output to the screen as it is produced. You will see all output resulting from a command displayed without interruption.

When $PAGING is set to TRUE, IPF 1100 displays a specified number of lines of output, and then either pauses or prints a message. In line mode, the number of lines of output displayed is determined by $PAGELENGTH. $PAGEWIDTH determines the number of columns displayed on each line. The value of $WAITTIME determines whether IPF 1100 pauses or prints a message asking you if you want to stop or continue the output.

The following is an example of entering a LIST command with the system variable $PAGING set to TRUE and $PAGELENGTH set to 21:

```
▸ ~C~▸LIST IPF*DIR.
▸-------------------------------------------------------------
▸ DIRECTORY OF IPF*DIR(1).
▸-------------------------------------------------------------
▸ NAME            TYPE    SUBTYPE
▸ ----            ----   ------
▸ ONE             SYM       SYM
▸ TWO             SYM       SYM
▸ THREE           SYM       SYM
▸ FOUR            SYM       SYM
▸ FIVE            SYM       SYM
▸ SIX             SYM       SYM
▸ SEVEN           SYM       SYM
▸ EIGHT           SYM       SYM
▸ NINE            SYM       SYM
▸ TEN             SYM       SYM
▸ ELEVEN          SYM       SYM
▸ TWELVE          SYM       SYM
▸ THIRTEEN        SYM       SYM
▸ FOURTEEN        SYM       SYM
▸ FIFTEEN         SYM       SYM
▸ SIXTEEN         SYM       SYM
▸Enter STOP to cancel output▸
```

After pressing XMIT, IPF 1100 continues to prompt you after every 21 lines of output until it has completely listed the directory.

IPF 1100 uses processors in order to execute some of its commands. These include COPY, DESCRIBE, ERASE, INVENTORY, LIBRARY, PACK, POSITION, and possibly others. IPF 1100 has no control over the output of the processor calls generated by these commands, but it attempts to determine the number of lines of output one of these commands produces. IPF 1100 issues a page break prior to one of these commands if the total number of lines displayed since the last page break is expected to exceed the value of $PAGELENGTH. Therefore, when using these commands, some "pages" can be shorter than $PAGELENGTH lines.

1.7.2. $WAITTIME

You can use the system variable $WAITTIME with $PAGING for a more flexible output environment. If $WAITTIME is 0 and $PAGING is set to TRUE, IPF 1100 displays a specified number of lines of output (determined by $PAGELENGTH) in line mode. A message then asks if you want to continue with the output or stop the output.

If you set $WAITTIME to a valid integer other than 0, IPF 1100 does not display the stop or continue message. Instead, IPF 1100 waits the specified number of seconds before displaying the next page of output.

1.7.3. $PAGECALL

$PAGECALL determines whether IPF 1100 pauses before or after it displays processor output. IPF 1100 cannot control output from other processors and programs. Therefore, if you generate output from a program by using the "CALL" command, IPF 1100 does not page the output. However, you can use the OUT command to redirect processor output.

Note: *In full-screen mode, the DESCRIBE and INVENTORY commands produce a prompt after all output has been generated. This prompt enables you to view the output before IPF 1100 returns to the full-screen display. However, if the DESCRIBE or INVENTORY command is used as part of a multiple command line or in a procedure, no prompt appears.*

IPF 1100 uses processors to execute some of its commands (such as, DESCRIBE and INVENTORY), and cannot page output from them. Therefore, if you want to pause before or after processor output, set $PAGECALL appropriately.

In full-screen mode, IPF 1100 stops after an IPF 1100 command, such as DESCRIBE or INVENTORY, that produces output generated by other processors even if $PAGING is FALSE.

1.8. Using IPF 1100 Commands

IPF 1100 commands consist of an action word followed by a series of keyword parameters. These keyword parameters add flexibility to the commands. However, you do not have to specify many of these keyword parameters because many of them have default values. There are two types of keyword parameters: required and optional. In the command formats that follow, the required keyword parameters appear first. The optional keyword parameters are enclosed in brackets. (You do not type the brackets.) See About This Guide for notation conventions.

1.8.1. Command Format

IPF 1100 commands have this format

```
command-name keyword=user-supplied-value [  keyword=user-supplied-value]...
```

For example

```
CREATE FILE=ABC*COBOL. DEVICE_TYPE=F33
```

where:

CREATE
: Is the command-name

FILE and DEVICE_TYPE
: Are keywords

ABC*COBOL and F33
: Are user-supplied values for the filename and the device type, respectively

You can specify these keyword parameters in any order, separating them by one or more spaces. IPF 1100 uses default values for any optional keyword parameters you do not specify. The following two examples of the CREATE command are equivalent:

```
CREATE FILE=RKK. DEVICE_TYPE=F12 VOLUME=DISK2
CREATE DEVICE_TYPE=F12 VOLUME=DISK2 FILE=RKK.
```

1.8.2. Keyword Parameter Format

The format of all keyword parameters is

```
keyword=user-supplied-value
```

The keyword itself appears in this guide as all uppercase characters. Type the keyword, if used, exactly as shown.

The *user-supplied-value* appears in this guide as all italicized lowercase characters.

Here is an example of an IPF 1100 command format:

```
CREATE FILE=filename
```

Therefore, you could actually enter the following command, where you supply the actual value ABC*COBOL. for the user-supplied value *filename*.

```
CREATE FILE=ABC*COBOL.
```

Use one or more spaces as the separator between the command name and the keyword parameters as well as between the keyword parameters themselves. Multiple spaces have the same meaning as a single space. In the construct *keyword=user-supplied-value*, spaces may occur between the keyword and the equal sign, and between the equal sign and the *user-supplied-value*, but are not necessary.

1.8.3. Command and Keyword Abbreviations

You can abbreviate most IPF 1100 command verbs and keywords to their first four characters. For keywords that are composed of two words joined together by an underscore character (_), type the first four characters of each (separated by _) if you want to abbreviate them. Note that this is an underscore (_), not a hyphen (-).

Here are some examples:

```
CREATE        CREA
POSITION      POSI
DEVICE_TYPE   DEVI_TYPE
```

You can use three-character abbreviations for some commands. These abbreviations are shown with the command in Section 7.

You can also abbreviate to their first four characters standard user-supplied values (i.e., if they are one of a standard set of reserved values). For example, abbreviate TEMPORARY to TEMP:

```
CREA FILE=A.  LIFE=TEMP
```

You can type either lowercase or uppercase characters or mix them together for IPF 1100 commands, keywords, and filenames. If something you type must be exactly as you type it, enclose it in quotation marks (''). These commands are equivalent:

```
CREATE FILE=BASIC*FILE-FILE.
create file=basic*file-file.
```

The *user-supplied-value* names have numbers appended to them where they appear more than once in a format. For example:

```
COPY FROM=filename-1 TO=filename-2
```

You can specify a number of keyword parameters after CREATE FILE= *filename*. These keyword parameters let you specify a specific storage device, the initial and maximum size of the file, and so forth.

1.9. Format Conventions

In addition to the conventions explained in About This Guide, IPF 1100 commands use the conventions described in the following paragraphs. Here are some general conventions:

- If you need to include a space as part of a user-supplied value, enclose this value in the delimiter characters (the value of the system variable $DELIMCHAR, initially quotation marks). For example, "my name".
- Most user-supplied values may be expressions (see 1.9.9).
- IPF 1100 uses default values for most keyword parameters when you do not specify values.

1.9.1. Comments

To include comments in IPF 1100 command lines, use the at symbol (@). An @ may not appear in column one. IPF 1100 considers anything to the right of an @ in an IPF 1100 command line a comment. For example:

```
CREATE FILE=MBS*COBOL. @ This creates a new file
                     @ for my COBOL programs
```

You can change the comment character by changing the value of the system variable $COMMENTCHAR (see 5.2).

1.9.2. Continuation Lines

To continue an IPF 1100 command line to the following line, use an ampersand (&). You can use an & in any character position where a blank character (space) is legal (except inside a string). The & must be the last nonblank character before a comment or the end of the line. For example:

```
CREATE FILE=&@This command continues on the next line
            MBS*COBOL.
```

You can change the continuation character by changing the value of the system variable $CONTCHAR (see 5.2).

1.9.3. Multiple Commands on One Line

You can specify more than one IPF 1100 command on the same line by separating them with semicolons. For example:

```
CREATE FILE=ABC*NEW.; PURGE FILE=DEF*OLD.
```

You can change the character that separates multiple commands on the same line by changing the value of the system variable $MULTICMDCHAR (see 5.2).

1.9.4. Identifiers

Identifiers include user or system variable names and label names. Identifiers can be up to 32 characters long, starting with a letter, followed by any sequence of letters, digits, and the underscore character (_). IPF 1100 does not distinguish between uppercase and lowercase letters in identifiers. Here are some examples of identifiers:

```
GRAND_TOTAL    K    B476253      Subtotal
```

Note: *You cannot use successive underscore characters. For example, AB_ _CD is not a valid identifier.*

1.9.5. Labels

Labels are any legal IPF 1100 identifier immediately followed by a colon. A label may precede any IPF 1100 procedures command. Here is an example of a label:

```
LAB1: CREATE FILE=B. LIFE=TEMP
```

where LAB1 is a label.

Labels are useful as reference points. They are usually used to change the order IPF 1100 executes commands in a procedure.

If you specify a label outside of a procedure, you will receive an error. For more information, see the *IPF 1100 Procedures User's Guide*.

1.9.6. Constants

A constant is a number with a value between $-(2^{35}-1)$ and $2^{35}-1$. It may have a decimal point and may contain up to 10 digits after the decimal point. It may have a + or - sign preceding it. It may not have any commas. Here are some examples of constants:

```
356   4.6784  -58   52     -4.982301
```

1.9.7. Conditions

A condition is one of these words:

```
TRUE    YES   ON    (these are equivalent)
FALSE   NO    OFF   (these are equivalent)
```

IPF 1100 stores these conditions internally as numeric 1 for TRUE and 0 (zero) for FALSE.

1.9.8. Strings

A string consists of any contiguous sequence of characters (including blanks but excluding quotation marks). There are two types of strings:

- Quoted - zero or more characters enclosed by the delimiter character (the value of the system variable $DELIMCHAR, which is initially a quotation mark "). If you need to use the delimiter character itself in a string, use two (or change the value of $DELIMCHAR). Examples of quoted strings are "Enter a number", "15+6=blue", "154 Park Avenue".
- Unquoted - a sequence of one or more characters not enclosed by the delimiter character. You can use unquoted strings in both string expressions and in full expressions. See 1.9.9 for more information.

1.9.9. Expressions

You can combine user and system variables, constants, and strings with various operators to form expressions. IPF 1100 evaluates an expression and returns a value. You can use expressions to supply values for many of the keyword parameters in IPF 1100.

There are two types of expressions: string and full. String expressions return results requiring a string such as filenames or choices and allow more characters to be unquoted. Full expressions return a number, string, or condition value as required by a parameter's definition.

String Expressions

You can concatenate (that is, join together) strings, variables, and functions with ampersands (&) to form string expressions.

A string expression consists of

```
{ string | variable |  function} & {string | variable |  function }
```

String expressions are very useful in ACCEPT commands. For example:

```
ACCEPT %X PROMPT="Today is " & $DATE ". What day do you wish to start on?"
```

The string concatenation operator (&) requires that the operands be strings. The result is a string formed by appending the second string to the first.

An unquoted string may not begin with these characters: % $ ()

For a string expression, an unquoted string may contain alphabetic characters, numeric characters, or any one of the following special characters:

```
* + - () / _ : . $ % # @ [] \ ^ ` {} ~ | !
```

If you use parentheses, you must have a right parenthesis for every left parenthesis.

An unquoted string may not contain the current values of these system variables: $COMMENTCHAR, $CONTCHAR, $DELIMCHAR, $MULTICMDCHAR.

Full Expressions

The format of a full expression is

```
[unary-operator] operand [binary-operator [unary-operator] operand]...
```

An operand can be a constant, string, condition, user variable, system variable, system function, or another full expression enclosed within parentheses.

You can combine operands with the operators that follow to form a full expression. Full expressions return a value. Whether this value is numeric, TRUE or FALSE, or a string depends on the type of expression and where you use it. Just like string expressions, you can use full expressions as part of IPF 1100 commands.

1.9.10. Unary and Binary Operators

These are the unary operators:

+	arithmetic identity
-	arithmetic identity
NOT	Boolean complement

These are the binary operators:

*	arithmetic product
/	arithmetic quotient
+	arithmetic sum
-	arithmetic difference
&	string concatenation
<	relational less than
<=	relational less than or equal to
<>	relational not equal
==	relational equals
>	relational greater than
>=	relational greater than or equal to
AND	Boolean product
OR	Boolean sum

The binary operators appear in decreasing order of precedence. You can use parentheses to group together pieces of an expression. All binary operators group left to right.

The arithmetic operators (*,/,+,-) require numeric operands and return a numeric result.

The string concatenation operator (&) requires that the operands be strings. The result is a string formed by appending the second string on to the first string.

The relational operators (==,<>,<,>,<=,>=) compare two values and return a condition as their result. If both values are strings, IPF 1100 uses string comparison. If $CASESENSITIVE is set to FALSE, the comparison is case insensitive, otherwise comparison is case sensitive. If both values are not strings, IPF 1100 converts the values to numbers and compares numerically.

The Boolean operators (AND, OR, and NOT) require condition operands. The operator AND returns TRUE if both operands are TRUE, otherwise it returns FALSE. The operator OR returns TRUE if either operand is TRUE, otherwise it returns FALSE. The operator NOT returns TRUE if its operand is FALSE and returns FALSE if its operand is TRUE.

If the operands for a given operator are the wrong type, IPF 1100 attempts to convert them. For example, if you try to add two strings, IPF 1100 tries to change them into numbers and proceeds with adding them.

If you use a real number where IPF 1100 expects an integer value, IPF 1100 truncates the real number to an integer.

For a full expression, an unquoted string may contain only letters, digits, and underscores. It must contain at least one letter or underscore. You can never leave the following strings unquoted since they are also operators:

```
AND OR NOT
```

1.9.11. Variables

IPF 1100 automatically converts the value of a variable to match the type of expression you are using. If your expression is a string expression and you use a variable containing a number, IPF 1100 will change its value into an equivalent string. If the expression is numeric and you use a variable containing a string, IPF 1100 will try to change its value into a number.

1.9.12. Lists

Some commands allow you to specify more than one value for a keyword parameter. This is known as a list. A list contains one or more integers, real numbers, strings, or expressions. If you specify more than one value, you must separate them with commas and enclose the list in parentheses.

If a keyword parameter requires a value and you enter a given choice more than once, IPF 1100 ignores the duplication.

IPF 1100 stops processing a command containing a list if any entry in the list is syntactically incorrect. If all entries are syntactically valid, IPF 1100 executes the command for each entry in the list.

1.10. Using Positional Notation

IPF 1100 has an inherent positional notation for each command; it recognizes the order in which you specify some keyword parameters. When you preserve this positional order, you may drop the keyword= from a field and separate the fields with commas. In the following example, the first command uses keyword notation. The second command is the same command using positional notation.

```
CREATE FILE=FILE1. DEVICE_TYPE=F33 VOLUME=DISKR4
CREATE FILE1.,F33,DISKR4
```

If you skip a field when using positional notation, you must preserve its position with a comma (for example, CREA RKK.,,DISK2). Spaces between commas, before commas, or after commas are not significant.

You can mix keyword notation and positional notation, but once you use keyword= notation, you must use it for the remainder of the command. If you have skipped a field using positional notation, you cannot go back to it by using keyword notation.

Therefore, the following example is legal because the keyword parameter FILE= is the last field you specified by position, allowing you to specify VOLUME and DEVICE_TYPE in any order using keyword notation.

```
CREA RKK. VOLUME=DISK2  DEVICE_TYPE=F12
```

However, the following example is not legal because once you use keyword notation, you must continue to use it to the end of the command.

```
CREA RKK. DEVICE_TYPE=F12 DISK2
```

The following example is also illegal because the keyword DEVICE-TYPE specifies a field passed over in the positional notation:

```
CREA RKK.,,DISK2 DEVICE_TYPE=F12
```

You can specify many keyword parameters by position. The keyword parameter description for each command describes the position assigned to it. See Section 7. Here are some rules for using positional and keyword notation:

- With keyword notation, you can specify keywords with their values in any order.
- With positional notation, if you skip a keyword parameter, you must preserve its position with a comma. For example, CREATE FILE1.,,DISKR4
- Once you use keyword notation (i.e., *keyword=user-supplied-value*), you must use it for the remainder of the command.
- The keyword parameter description for each command in this manual indicates whether you can specify it by position, and if so, the position assigned to it.

1.11. Calling and Leaving IPF 1100

To call IPF 1100, type the following call after a start of entry character (hereafter called the prompt) and press XMIT:

```
▸@IPF▒
```

IPF 1100 responds with this message, where *mm/dd/yy* is the current date in month, day, year format and *hh:mm:ss* is the current time in hour, minute, and second format:

```
▸IPF 1100 7R1 mm/dd/yy hh:mm:ss
```

IPF 1100 then solicits commands using the prompt:

```
▸ ~C~▸▒
```

For example, enter ▸@IPF. The response should look like this simulated screen display.

```
▸@IPF
▸IPF 11006R3 09/23/96 11:44:23
▸ ~C~▸▒
```

To leave IPF 1100, enter

```
▸ ~C~▸LOGOFF ▒
```

IPF 1100 responds with the following message and ends your IPF 1100 session:

```
▸END IPF
```

1.12. Signing Off Your System

If you want to sign off your computer system, enter

```
▸@FIN ▒
```

You will receive an accounting summary on your screen similar to this:

```
RUNID: ABC ACCT: 425520 PROJECT: FINAN
ABC FIN
TIME: TOTAL: 00:05015.268          CBSUPS: 011992517
CPU: 00:00:00.787                  I/O:00:00:11.498
CC/ER: 00:01:03.567                WAIT: 00:00:37.677
SUAS USED:4.00                     SUAS REMAINING:2000.00
IMAGES READ:39     PAGES: 2
START:  11:44:23 SEP 23,1996  FIN: 11:45:35 SEP 23,1996
*TERMINAL INACTIVE*
```

Section 2
Directories and Files

This section presents the concepts of IPF 1100 directories and files. This section also points out an important distinction: the difference between a data image and a screen line.

2.1. Working with Directories and Files

A file is an area of storage on the system that contains data: a computer program, a document database, a group of related data records, etc.

You can save files that have some logical relation to each other within a directory. Think of your computer system as a big filing cabinet. The filing cabinet drawers represent directories.

The file folders within each drawer represent files within a directory.

Going one step further, because the file folder represents a file, the paper within the file folder represents the data within a file.

2.2. Naming Files

Many IPF 1100 commands require you to supply a filename. This can be either the name of a directory, or a specific file in the directory. If you specify just the name of a directory, you imply all of the files within the directory.

Each file on your system has a unique name to distinguish it from all other files. Filenames can have these components:

```
NAME*COLOR.BLUE
```

where:

NAME	Is the qualifier, which is followed by an asterisk. If you omit the qualifier, IPF 1100 uses your project-id.
COLOR.	Is the name of a directory (or data file), which is followed by a period.
BLUE	Is the name of a file (unless you are using a data file).

All three of these components can be up to 12 characters from the set A-Z, 0-9, and $. The qualifier is another name you can specify to avoid conflicts if you have two directories (or data files) with the same name.

To make sure you have a unique filename, you may specify additional components to compose a fully qualified filename. The expanded filename format is

*qualifier*directory-name(cycle)/read-key/write-key.file/version-name*

The following subsections describe the additional information you can use.

2.2.1. Using Read and Write Keys

Read and write keys may contain up to six characters, not including blank, comma, slash, period, or semicolon. Keys are optional and specify whether the file is a read- or write-only file.

Once IPF 1100 attaches a file to your run, it ignores read and write keys on subsequent references of the file. Many of the IPF 1100 commands attach your file for you. If you wish to specify keys on a filename that IPF 1100 previously attached without keys, free the file before entering your command with keys.

For example, assume the file COLOR exists and was created with a write key, but is not attached to your run.

```
▸ ~C~▸OLD COLOR. @OLD command will attach the file to your run.
▸ ~C~▸10 Enter a line.
▸ ~C~▸REPLACE COLOR//w. @the write key is ignored.
▸**WORKFILE180 You cannot update file COLOR.
▸ ~C~▸FREE COLOR.   @free the file so you can supply the correct keys.
▸ ~C~▸REPLACE COLOR//w.
```

You should include the correct key or keys the first time you refer to the file. The following is more efficient than the previous example:

```
▸ ~C~▸OLD COLOR//w.
▸ ~C~▸10 Enter a line.
▸ ~C~▸REPLACE @There is no further need to enter its keys.
```

Many IPF 1100 commands do not work on files that are attached in a read-inhibited state. This is because these commands need to be able to look at the file during their processing of it.

2.2.2. Using Cycles

The cycle may be an integer in the range -31 to +1 (relative) and 0 to 999 (absolute). This number differentiates this directory from a set of related directories with the same name.

When you specify a filename without specifying a cycle, IPF 1100 uses the most recently created cycle of the file.

For example:

```
CREATE COLOR(1).
CREATE COLOR(2).
CREATE COLOR(3).
ATTACH COLOR. @attaches cycle 3 to your run
```

2.2.3. Using Version Names

The *version-name* is from 1 to 12 characters from the set A-Z, 0-9, hyphen, or $.

By using version names, you can have many files with the same name, but with different version names. For example:

```
GJH*DIRECNAME.FILENAME/VER-ONE
GJH*DIRECNAME.FILENAME/VER-TWO
GJH*DIRECNAME.FILENAME/VER-THREE
GJH*DIRECNAME.FILENAME/VER-FOUR
GJH*DIRECNAME.FILENAME/VER-FIVE
```

These are all legal IPF 1100 filenames:

```
ABC*DEF.
PAYROLL
LUNAR*ONE.TYCHO
```

These are not legal IPF 1100 filenames:

```
#HOLD.IT  (illegal # character)
@56*8FD.  (illegal @ character)
```

Note: *When specifying filenames on a remote host, a host-id must precede the filename. See the* DDP-FJT IPF and Batch Interface Operations Guide.

2.3. Using Data Files

For certain uses, you may need to have a file not associated with (that is, not within) a directory. This is true when you are using tape files or print files (see 2.13 and 2.20).

```
MY*DATAFILE.
```

Here MY*DATAFILE. is a data file. A data file always has a filename ending with a period with nothing following it. Although a data file can have the same data in it as a file which is contained within a directory, a data file is independent of any directory (i.e., it stands alone).

2.4. Assigning a Home Directory

Each time you sign on to IPF 1100, it assigns you a home directory. This directory contains the files you use most often during your IPF 1100 session.

You can specify your home directory when you call IPF 1100 by using the following command:

```
▸@IPF home-directory.
```

For example:

```
▸@IPF JLK*ACC.
```

If the home directory you specify does not exist, IPF 1100 creates it after displaying the message:

```
Your HOME directory does not exist.  It will be created.
```

If you do not specify a home directory, IPF 1100 assigns one to you with the name:

```
project-id*user-id.
```

IPF 1100 takes the *user-id* entry and the *project-id* from the @RUN statement you entered (or your system generated for you) at the time you signed on. For example:

```
▸ENTER USERID/PASSWORD CLEARANCE LEVEL
▸*mdw/mypass
.
.
.
▸@RUN acc,431340,budget
▸DATE: 080393 TIME: 082234
▸@IPF
▸IPF 11006R1 08/03/94 08:22:38
▸ ~C~▸
```

The name of the home directory in this example is "budget*mdw." When IPF 1100 supplies a value for something when you do not specify one, we call that a default value.

Notes:

- *If your site does not use user-ids as part of the sign-on procedure, IPF 1100 will use your run-id instead of your user-id in constructing a default name for your home directory (project-id*run-id). In the preceding example, your home directory would be "budget*acc."*
- *Your home directory is reserved for use by IPF 1100 and you should not directly manipulate it (for example, by using the FREE or PURGE command). IPF 1100 may produce unpredictable results if you directly manipulate your home directory.*

2.5. Defining Work Directories

You also have a directory called the work directory. When you sign on to IPF 1100, your work directory and home directory are the same. Unlike the home directory, which you cannot change, you can change the work directory by setting the system variable $WORKDIRECTORY. You can display the name of your home and work directories by displaying the system variables $HOMEDIRECTORY and $WORKDIRECTORY.

IPF 1100 automatically searches your work directory if you specify a filename without a directory name on commands, such as

OLD	ERROR
NEW	OUT
SAVE	IN
REPLACE	LIST
COPY	NAME
CREATE	PURGE
DESCRIBE	

When you enter an IPF 1100 command requiring the name of a file, but omit a directory name, IPF 1100 looks in your work directory for the file. For example, suppose you call IPF 1100 with

```
▸@IPF MY*HOME-DIR.
```

The OLD command retrieves an existing file. If you enter the following command, IPF 1100 searches your work directory MY*HOME-DIR. until it finds a file with the name GREEN.

```
▸ ~C~▸OLD FILE=GREEN
```

But, if you enter the following command, IPF 1100 knows you are specifying the file GREEN within the directory MY*DIR.

```
▸ ~C~▸OLD FILE=MY*DIR.GREEN
```

Put the files that you want always available in your work directory.

System directories are also available to all users on your system. They contain commonly used programs, language compilers, and other processors.

Note: *Your work directory is reserved for use by IPF 1100 and you should not directly manipulate it (for example, by using the FREE or PURGE command). IPF 1100 may produce unpredictable results if you directly manipulate your work directory.*

2.6. Data Images and Screen Lines

There is an important distinction between data images and screen lines. A data image is a variable-length string of characters in your file (currently limited to a maximum of 256 characters). A screen line is one display line on your terminal screen (up to 80 characters). Because you may have data images longer than a screen line, IPF 1100 may display one data image on more than one screen line.

The maximum length of a screen line is generally 80 columns (that is, character spaces). This may vary, depending on the particular display terminal you use. We usually call a data image an image, and a screen line a line.

2.7. Creating Files and Directories

You must create a directory (or data file) before you save any new data. Use the CREATE command to create a new directory, data file, or file within a directory on your system. Its basic format is

```
CREATE FILE=filename
```

You must specify a *filename* (that is, the name by which you will call the file). After IPF 1100 processes this command, it catalogs a file with the filename you specified.

Cataloging means that IPF 1100 recognizes the *filename* in future sessions and saves the file's contents from one IPF 1100 session to the next. In other words, it reserves some storage on your system. For example:

```
▸@IPF
▸IPF 1100 6R1 08/03/94 09:43:56
▸ ~C~▸CREATE FILE=MY*FILE.
```

This example created a file named MY*FILE. Whether the file is a directory or a data file is not decided until you start using it.

You can create a directory, a data file, or a file within a directory. You can only create a file within a directory if the directory already exists. For example, if you have a directory "A", you can enter CREATE FILE=A.B to create an empty file "B". If the directory does not exist, IPF 1100 displays an error message.

If you want to create files on magnetic tape, see 2.21.

2.8. Creating Public and Private Files

You can specify a number of keyword parameters after CREATE FILE=filename. These keyword parameters let you specify a specific storage device, the initial and maximum size of the file, and so forth.

One important keyword parameter is ACCESS=. If you enter CREATE FILE= *filename* ACCESS=PUBLIC, the file you create is a public file and any user can access your file.

If you enter CREATE FILE=*filename* ACCESS=PRIVATE, the file is a private file and other users must sign on using the same project-id to access your file. If you do not specify ACCESS= (just enter CREATE FILE= *filename*), IPF 1100 uses a default of PRIVATE.

Here are some other examples of CREATE commands entered on a terminal and how IPF 1100 responds to them.

```
▶ ~C~▶CREATE FILE=ABC*ONE. ACCESS=PRIVATE      ①
▶ ~C~▶CREATE FILE=ABC*TWO.     ②
▶ ~C~▶CREATE FILE=ABC*ONE.     ③
▶**FACIL2180 Your file ABC*ONE. already exists.      ④
▶ ~C~▶CREATE FILE=ABC*THREE. ACCESS=PUBLIC       ⑤
▶ ~C~▶CREATE ABC*TWO.TODAY     ⑥
```

Explanation

① Create a private file named ABC*ONE. At this point, ABC*ONE. could be a directory or a data file depending on what you subsequently do with it.

② Create a new file named ABC*TWO. The file defaults to a private file.

③ Tell IPF 1100 to create another new file named ABC*ONE.

④ IPF 1100 tells you that a file by that name already exists on your system (you created a file by that name at 1).

⑤ Create a new file named ABC*THREE. Since you specify that you want the file to be public (ACCESS=PUBLIC), other IPF 1100 users can access your file.

⑥ Create an empty file named ABC*TWO.TODAY. Note that you use positional notation, not typing FILE=. The directory ABC*TWO. already exists (you created it at ②).

For more information on the CREATE command, see 7.2.6.

2.9. Creating Temporary Files

If you want to create a temporary file (a file that will not exist after you leave IPF 1100 and end your terminal session with an @FIN command), use the keyword parameter LIFE=TEMPORARY on the CREATE command (the default for LIFE= is PERMANENT).

Here is an example of creating a temporary file:

```
▶@IPF
▶IPF 1100 7R1 03/27/96 08:10:12
▶ ~C~▶CREATE FILE=A.    ①
▶ ~C~▶CREATE FILE=B.  LIFE=TEMPORARY
▶ ~C~▶LOGOFF      ②
▶END IPF
▶@FIN
```

Explanation

① Create two files. File A. is a permanent file. File B. is a temporary file.

② Leave IPF 1100 and terminate your session with an @FIN command. File B. no longer exists.

For more information on the CREATE command, see 7.2.6.

2.10. Listing Files in a Directory

The LIST command tells IPF 1100 to display information on your terminal about the files within a directory (or one specific file within a directory). The format of the LIST command is

```
LIST FILE=filename
```

The *filename* must be the name of a directory or the name of a file within a directory. It cannot be a data file. IPF 1100 displays the names of the files in the directory as well as their types and subtypes. (Types and subtypes are described Section 4.)

If you just enter LIST, IPF 1100 displays information about your work directory. If you want more information about the files in the directory (the size as well as the last update time and date), enter LIST FILE=*filename*FORM=LONG.

Here are some examples of using the LIST command:

```
▸ ~C~▸LIST FILE=JYS*COLORS.      ①
▸------------------------------------------------------------------------
▸  DIRECTORY OF JYS*COLORS(1).      ②
▸------------------------------------------------------------------------
▸  NAME                    TYPE     SUBTYPE
▸  ----                    ----     ------
▸  BLUE                    SYM        SYM
▸  GREEN                   SYM        ELT
▸  YELLOW                  SYM        COB
▸  RED                     SYM        IPF
▸ ~C~▸LIST FILE=JYS*SHAPES. FORM=LONG      ③
▸------------------------------------------------------------------------
▸  DIRECTORY OF JYS*SHAPES(1)   ④
▸------------------------------------------------------------------------
▸  NAME                    TYPE     SUBTYPE   SIZE  LAST UPDATE  DATE/TIME
▸  ----                    ----     -------   ----  -------------------
▸  SQUARE                  SYM        SYM     160    08/15/92    14:05:29
▸  CIRCLE                  SYM        IPF     340    08/16/92    06:15:49
▸  DIAMOND                 SYM        COB     654    08/16/92    11:22:33
▸  TRIANGLE                SYM        IPF     120    08/16/92    14:02:02
▸  OVAL                    SYM        SYM     180    08/17/92    10:22:29
▸ ~C~▸
```

Explanation

① Ask for a listing of the directory JYS*COLORS.

② IPF 1100 displays the files included in the directory JYS*COLORS.

③ Ask for a listing of the directory JYS*SHAPES. and specify FORM=LONG.

④ IPF 1100 displays the names of the files with additional information (size, and date and time of the last update).

For more information on the LIST command, see 7.2.18.

2.11. Deleting a File

When you no longer need a file, delete it using the PURGE command. The format of the PURGE command is

```
PURGE FILE=filename
```

You must specify a filename. You can also specify the keyword parameter TYPE= with one or more of these values: SYMBOLIC, RELOCATABLE, ABSOLUTE, OMNIBUS, and ALL.

You can use this TYPE keyword parameter only for deleting files in a directory. IPF 1100 deletes the specified type of the file with the name *filename* from the system and rejects subsequent references to it. You can delete

- A directory. For example, PURGE FILE=INVENTORY.
- One ore more types of a file in a directory. For example, PURGE FILE=INVENTORY.BOLTS TYPE=(SYMB,RELO).
- A data file. For example, PURGE FILE=DATAFILE.

If you created the file MY*FILE. earlier, you can now delete it by entering

```
PURGE FILE=MY*FILE.
```

Here are some examples of using PURGE commands on a terminal and how IPF 1100 responds to them:

```
▸ ~C~▸PURGE FILE=ABC*DIREC.A      ①
▸ ~C~▸PURGE FILE=RJS*REPORT.ONE TYPE=(SYMB,ABSO)      ②
▸ ~C~▸PURGE FILE=OLD*DATAFILE.      ③
▸**FACIL2110 IPF 1100 did not find the file OLD*DATAFILE on the system.      ④
▸ ~C~▸PURGE FILE=ABC*DIREC.      ⑤
▸ ~C~▸PURGE X*Y.Z      ⑥
▸ ~C~▸PURGE FILE=CAB*RED.PM,ALL      ⑦
```

Explanation

① Tell IPF 1100 to delete a file named A in directory ABC*DIREC.

② Tell IPF 1100 to delete the symbolic and absolute file types of the RJS*REPORT.ONE file.

③ Tell IPF 1100 to delete a file with the name OLD*DATAFILE.

④ Because no file by this name exists, IPF 1100 displays this message.

⑤ Tell IPF 1100 to delete the directory ABC*DIREC. and all the files it contains.

⑥ Delete symbolic file X*Y.Z using positional notation.

⑦ Delete all existing file types of the file CAB*RED.PM.

For more information on the PURGE command, see 7.2.27.

2.12. Copying a File

Use the COPY command to copy data from one file or directory to another file or directory on your system. The format of the COPY command is

```
COPY FROM=filename-1 TO=filename-2
```

You must specify both *filename-1* and *filename-2*. Think of *filename-1* as the "FROM file" and *filename*-2 as the "TO file."

Before using a COPY command, make sure both the FROM and TO files (i.e., *filename-1* and *filename-2*) exist.

You can use the COPY command to copy

- Entire directories (for example, COPY FROM=A. TO=B.)
- Data files (for example, COPY FROM=DATA1. TO=DATA2.)
- One file of a directory to a file of another directory (for example, COPY FROM=A.1 TO=B.1)
- One file of a directory to another file of the same directory (for example, COPY FROM=A.1 TO=A.2)
- One file of a directory to a data file (for example, COPY FROM=A.1 TO=DATA1.)
- One data file to a file within a directory (for example, COPY FROM=DATA1. TO=A.1)

When copying data from one file to another, you can specify whether you want to save the original data already in the TO file. The keyword parameter POSITION= determines whether the contents of the FROM file replace the contents of the TO file (POSITION=REPLACE) or if IPF 1100 adds the FROM file to the contents of the TO file (POSITION=ADD). For example:

```
COPY FROM=FILE1. TO=FILE2. POSITION=REPLACE
```

If you do not specify a value for POSITION= on a COPY command, IPF 1100 defaults to POSITION=REPLACE.

Here are some examples of COPY commands and how IPF 1100 responds to them:

```
▶ ~C~▶COPYFROM=DIR*1. TO=DIR*2. POSITION=REPLACE       ①
▶ ~C~▶COPYFROM=DIR*A. TO=DIR*B.  POSITION=ADD       ②
▶ ~C~▶COPYFROM=ABC*D. TO=FILE*1.MASTER POSITION=REPLACE       ③
▶ ~C~▶COPYFROM=ABC*D. TO=NEW*FILE.       ④
▶**FILE111O IPF 1100 could not find the NEW*FILE. file       ⑤
▶ ~C~▶COPYDIR*1.A, DIR*2.B       ⑥
▶ ~C~▶COPYDIR*C.1, DIR*Z. POSITION=ADD       ⑦
▶ ~C~▶
```

Explanation

① Tell IPF 1100 to replace directory DIR*2. entirely with the contents of directory DIR*1.

② Copy directory DIR*A. to another directory, DIR*B. IPF 1100 adds the files of directory DIR*A. to directory DIR*B.

③ Tell IPF 1100 to copy the contents of a data file ABC*D. to a file MASTER within directory FILE*1. IPF 1100 does not alter any other files of FILE*1.

④ Copy a directory ABC*D. to NEW*FILE. In this instance, there is no data file or directory by the name of NEW*FILE on your system.

⑤ IPF 1100 displays an error message telling you that the TO file specified does not exist.

⑥ Enter a COPY command to copy a file DIR*1.A to a file DIR*2.B. Here, IPF 1100 is copying only one file (A) from the directory DIR*1. Also, you used positional notation (omitting the keywords FROM= and TO=).

⑦ Enter a COPY command to copy a file (1) included in a directory (DIR*C.) into another directory (DIR*Z.). By specifying POSITION=ADD, directory DIR*Z. retains its current set of files. The new file is named DIR*Z.1.

For more information on the COPY command, see 7.2.5.

2.13. Printing a File

When you want a paper printout of a specific file, use the COPY command. This is the same COPY command shown in 3.12 with some extra options.

Specify the name of the file you want printed after the keyword parameter FROM=, then specify the printer name (as known to your system) after the keyword parameter TO= and preceded by the qualifier DEV$. The format is

```
COPY FROM=filename.  TO=DEV$*printer.
```

The *filename* cannot be a directory name (it must be a data file or a file within a directory). The period following *printer*is optional.

Always precede *printer* with DEV$*. For example, to print a copy of the contents of the data file FILE1. on the high-speed printer configured at your site as PR2, enter

```
COPY FROM=FILE1.  TO=DEV$*PR2.
```

You can specify how many copies of the printout you want. To specify multiple copies, enter

```
COPY FROM=filename.  TO=DEV$*printerCOUNT=n
```

where *n* is the number of copies you want printed.

You can also specify whether or not (YES or NO) you want *filename* to remain cataloged on your system after printing it. To delete the file from your system after printing, enter

```
COPY FROM=filename.  TO=DEV$*printerRETAIN=NO
```

The default (RETAIN=YES) is for retaining the file on the system after IPF 1100 prints it. You can use the RETAIN= and COUNT= options together on the same COPY command.

Here are some more examples of using the COPY command to print files:

```
▶ ~C~▶OLD DATA*FILE.      ①
▶ ~C~▶PRINT ALL      ②
▶0010.00 red
▶0020.00 green
▶0030.00 blue
▶ ~C~▶COPY  FROM=DATA*FILE.TO=DEV$*PR1      ③
▶ ~C~▶COPY  FROM=DATA*FILE.TO=DEV$*PR2 COUNT=3 RETAIN=NO      ④
▶ ~C~▶COPY  FROM=DIR*FILE.A TO=DEV$*PR1      ⑤
▶ ~C~▶COPY  FROM=DIR*FILE.B TO=DEV$*PR2 COUNT=5 RETAIN=NO      ⑥
▶ ~C~▶
```

Explanation

① An OLD command retrieves the data file named DATA*FILE. to your workspace.

② You display the three images in your workspace.

③ The COPY command specifies printing the file DATA*FILE. on the printer identified to your system as PR1.

④ The COPY command specifies printing three copies of the file DATA*FILE. on the printer identified to your system as PR2. You also specify that IPF 1100 should delete the file after printing it.

⑤ The COPY command specifies printing a file (A) included in a directory (DIR*FILE.) on the printer identified to your system as PR1.

⑥ The COPY command specifies printing five copies of the file B included in directory DIR*FILE. on the printer identified to your system as PR2. You also specify that IPF 1100 should delete the file B after printing it.

2.14. Maintaining Files

There are several other IPF 1100 commands that you can use to perform file maintenance on data files and directories.

2.14.1. PACK Command

As you copy files in a given directory with SAVE and REPLACE commands, the directory begins to fill up. The old copies of the symbolic files still exist in the directory, even though you cannot look at them anymore. Eventually, you may get a message that there is not enough room in the directory to perform the SAVE or REPLACE command you requested.

To remove the old files from a directory, use the PACK command. This allows you to reuse as much of the directory as possible. It is also good practice to occasionally remove these old files with a PACK command to improve the handling of the files within the directory. The format of the PACK command is

```
PACK FILE=directory-name-list.
```

If you do not enter a directory name, the work directory is assumed.

Because IPF 1100 packs directories only, entering an individual filename or a data file on the PACK command causes an error message to appear.

For more information on the PACK command, see 7.2.25.

2.14.2. ERASE Command

Sometimes, you may want to completely remove all the files from a directory or to erase a data file. The ERASE command does this for you. Its format is

```
ERASE FILE=filename-list.
```

The filename must be the name of a directory or a data file. Specifying an individual filename causes an error message to appear. You must specify a filename (there is no default value).

This command releases all the files in a directory or all the data in a data file. It makes the directory or data file available for reuse.

For more information on the ERASE command, see 7.2.11.

2.15. Specifying an Additional Name for a File

It is often more convenient to use a shorter, temporary name instead of the long name of a file. Rather than type a long filename every time you have to specify the file, you enter a USE command to associate your own choice of a name (or identifier) with the actual filename.

The format of the USE command is

```
USE NAME=identifier FILE=filename
```

For example, if the actual filename is quite long, specify a shorter identifier:

```
USE NAME=N17 FILE=NOVEMBER*SEVENTEENTH.
```

You can specify an identifier of up to 12 characters. The period at the end is optional. The *filename* must be the name of a directory or a data file (but not a file within a directory).

You can use an identifier until either you finish your session, you assign the identifier to another file, or you release the file.

Here are some examples of USE commands:

```
▶ ~C~▶USE NAME=STAR. FILE=MILKYWAY*BETELGEUSE.      ①
▶ ~C~▶LIST  FILE=STAR.     ②
▶------------------------------------------------------------------------------
▶ DIRECTORY OF MILKYWAY*BETELGEUSE(1).
▶------------------------------------------------------------------------------
▶ NAME            TYPE  SUBTYPE
▶ ----            ----  ------
▶ REDGIANT        SYM     SYM
▶ INORION         SYM     SYM
▶ ~C~▶USE NAME=STAR. FILE=MILKYWAY*POLARIS.      ③
▶ ~C~▶LIST FILE=STAR.     ④
▶------------------------------------------------------------------------------
▶ DIRECTORY OF MILKYWAY*POLARIS(1).
▶------------------------------------------------------------------------------
▶ NAME            TYPE  SUBTYPE
▶ ----            ----  ------
▶ NEWBLUE         SYM     SYM
▶ INURSAMINOR     SYM     SYM
▶ LITTLEDIPPER    SYM     SYM
▶ ~C~▶
```

Explanation

① Tell IPF 1100 you will refer to the file MILKYWAY*BETELGEUSE by the identifier STAR.

② Display the files for directory STAR (MILKYWAY*BETELGEUSE). IPF 1100 always displays the fully qualified filename, even after you enter a USE command.

③ Tell IPF 1100 that you will refer to the file MILKYWAY*POLARIS by the identifier STAR. IPF 1100 disassociates the identifier STAR from the file MILKYWAY*BETELGEUSE and now associates it with the file MILKYWAY*POLARIS instead.

④ Enter another LIST command using the new identifier. IPF 1100 displays the directory for the file MILKYWAY*POLARIS.

For more information on the USE command, see 7.2.35.

2.16. Explicitly Assigning a File

IPF 1100 usually assigns and frees files as it needs them (that is, implicitly). For example, when you tell IPF 1100 to list the files in a directory, IPF 1100 takes care of assigning the directory. Assigning means that IPF 1100 can now access the file.

You must explicitly assign a file if you use

- Read or write keys on your files
- Cataloged tape files (see 2.21)

The ATTACH command provides this function. Its format is

```
ATTACH FILE=filename.
```

Here are some examples of the ATTACH command:

```
▶ ~C~▶ATTACH FILE=ACC*NOV.      ①
▶ ~C~▶ATTACH FILE=MY*CAT-TAPE.      ②
▶ ~C~▶ATTACH FILE=ACC*UTIL.      ③
▶**FILE6150 The file ACC*UTIL. is rolled out.      ④
▶ ~C~▶ATTACH FILE=ACC*MYFILE. ACTION=EXCLUSIVE      ⑤
▶ ~C~▶ATTACH FILE=ACC*WORK. ACTION=(WAIT,EXCLUSIVE)      ⑥
▶IPF 1100 started waiting at 11:08:36 for ACC*WORK. to be loaded from tape.  ⑦
▶@@X C      ⑧
▶**IPF6050 You have successfully terminated your ATTACH command.      ⑨
▶ ~C~▶
```

Explanation

① Explicitly assign a file ACC*NOV. IPF 1100 assigns this file to your job.

② Assign a cataloged tape file MY*CAT-TAPE.

③ Assign a file ACC*UTIL.

④ IPF 1100 displays a message telling you the latter file is not available now (you should try again later). This can happen on a system if you have not used your file in some time.

⑤ Assign file ACC*MYFILE. IPF 1100 exclusively assigns this file to your job, if it is available. Once assigned to you, no one else will be able to access it.

⑥ Explicitly assign file ACC*WORK. specifying that you want to wait for the file to be assigned exclusively to you.

⑦ IPF 1100 places your job in a wait state until file ACC*WORK. is available to be exclusively assigned to your job. An information message appears.

⑧ Terminate the wait state by entering ▶@@X C.

⑨ IPF 1100 displays an informational message when the wait condition is terminated.

You can also assign a specific file within a directory (for example, ATTACH FILE=MY*DIREC.FILE1). Doing this assigns the entire directory, provided the specific file exists.

The ATTACH command has an optional ACTION keyword parameter. You can specify one or more of these values for ACTION: NONE, WAIT, EXCLUSIVE, PURGE, RECOVER, or EXISTENCE.

However, you may not specify the ACTION keyword parameter when attaching a specific file within a directory because the special action then applies to the entire directory.

For more information on the ATTACH command, see 7.2.2.

2.17. Explicitly Releasing a File

Sometimes you may need to explicitly free the assignment of a file (for example, someone else may want to use it). Use the FREE command to do this. Its format is

```
FREE FILE=filename-list.
```

For example, if you previously entered an ATTACH command to assign a file and you no longer need it, enter a FREE command specifying the filename.

Here are some examples of the FREE command:

```
▸ ~C~▸ATTACH FILE=ACC*MASTER.      ①
▸ ~C~▸RUN ACC*MASTER.UPDATE        ②
▸Master File update now complete.      ③
▸ ~C~▸FREEFILE=ACC*MASTER.         ④
▸ ~C~▸
```

Explanation

① Explicitly assign a file ACC*MASTER.

② Tell IPF 1100 to run a (symbolic) program named ACC*MASTER.UPDATE.

③ When the program is finished, it displays a message telling you. (Note that a statement in your program displays this message, not an IPF 1100 command.)

④ Enter a FREE command, telling IPF 1100 that you no longer need the directory ACC*MASTER.

You can also indicate a specific action to be taken when IPF 1100 frees your file. For more information on the FREE command, see 7.2.14.

2.18. Retrieving Information about Files

The DESCRIBE command tells IPF 1100 to display information on your terminal about the directories or data files you specify. The format of the DESCRIBE command is

```
DESCRIBE FILE=filename-list
```

The *filename-list* can contain directory names, data filenames, or both. IPF 1100 displays the following information:

- Project identity of your run
- Account number of your run
- External name of the file
- Size of storage allocated for the file
- Access modes for the file
- Portion of file storage attached
- Date file was created
- Date of last reference
- Backup information

If you just enter DESCRIBE, IPF 1100 displays the information for your work directory.

Here is an example of using the DESCRIBE command:

```
▸ ~C~▸DESCRIBE FILE=JOR*WORK.
▸FURPUR 30R5 S75 08/20/94 14:45:30
▸ * * PROJ: JORACCNT: 529009
▸JOR*WORK(1),F/0/TRK/128
▸MODES: PRIVAT,ASG-D
▸NO. OF GRANULES ASG-D:0
▸HIGHEST GRANULE ASG-D:0TOTAL ASSIGNMENTS: 0
▸HIGHEST TRACK WRITTEN:0
▸CAT: 08/20/94 AT 14:17:20, LAST REF: 08/20/94 AT 14:17:24
▸ ~C~▸
```

You can also use the OUT command (see 6.4) to redirect your output to a file. You can later scroll through this file to view your output. See 1.7 for information on paging of your output.

For more information on the DESCRIBE command, see 7.2.7.

2.19. Listing Files Associated with a Session

Sometime during your IPF 1100 session, you may want to find out what files are associated with your run or the project-id of your run.

The INVENTORY command lets you obtain a list of files associated with your run or the project-id of your run. The format of the INVENTORY command is

```
INVENTORY SELECT=ATTACHED
```

or

```
INVENTORY SELECT=PROJECT_ID
```

If you enter INVENTORY SELECT=ATTACHED, IPF 1100 displays the names of all catalogued and temporary files currently attached to your run. If you enter SELECT=PROJECT_ID, IPF 1100 displays all files catalogued with the project-id of your run.

For example:

```
▸ ~C~▸INVENTORY SELECT=ATTACHED
▸FURPUR 30R5 S75 09/06/96 11:11:13
▸ABC*TPF$(0),D8470,T
▸ABC*DIAG$(0),D8470,T
▸SYS$*LIB$(1),D8470,A
▸SYS$LIB$*IPFCONT(3),D8470,A CSINTNAME$,
▸ABC*PSC(1),D8470,ZA IP$IPF$$DDS$,IP$USEAAAAAA,
▸SYS$LIB$*FURPUR(15),D8470,A CSINTNALT$,
▸ABC*IPF$PSC(1),D8470,ZXAIP$USEAAAAAB,
▸SYS$*DATA$(483),D8470,ZAIP$USEAAAAAC,IP$DATA$$$$$,IP$INTERNAL,
▸SYS$LIB$*IPF$PROC$(0),NAME ITEM IP$PROC$$$$$,
▸ABC*IP$ADDFILE(0),D8470,T @@@]1ZCPTK$$$,
▸ ~C~▸
```

Often the list of names is long and the output from the INVENTORY command scrolls off the screen before you can read it. If that happens, use the OUT command (see 6.4) to redirect your output to a file. You can later scroll through this file to view your output. See 1.7 for information on paging of your output.

For more information on the INVENTORY command, see 7.2.16.

2.20. Working with Tapes

Sometimes you will need to use magnetic tapes. You may be saving files for future use, transferring data to tape for use at a different installation, or you may be putting data on tape for processing by other equipment. IPF 1100 saves files on tapes sequentially.

There are three common tape operations that involve transferring data:

- Copying a file from the system (a mass-storage file) to a tape
- Copying a file from a tape to the system
- Copying data from one tape to another tape

Tape files must be entire directories or data files (that is, not files within directories). Using tapes involves three operations:

1. Mounting a tape on a tape drive (for example, a UNISERVO 28).
2. If necessary, positioning the tape to a specific point.
3. Copying data to or from the tape.

Tape files can be temporary or cataloged. Temporary only means that a tape file is not cataloged; however, the data on the tapes remains on them after your session is over.

2.20.1. Mounting a Tape

To mount a tape on a tape drive, enter

```
MOUNT FILE=filename.
      VOLUME=vol
      DEVICE_CLASS=TAPE
```

Although the format shows the FILE=, VOLUME=, and DEVICE_CLASS= keyword parameters on separate lines, you can type them on the same line on your screen, and you can specify them in any order.

The *filename* is any name unique to your session that you want to call the tape. The *vol* is an external number, usually on a label attached to the plastic housing of the tape. This number tells the computer operator which tape to mount on the tape drive. The *filename*associates a tag with that tape so you can subsequently refer to it.

For example, if you enter the following command, the system operator mounts your tape numbered 210 on a tape drive, and you subsequently refer to the tape as MY*TAPE.

```
MOUNT FILE=MY*TAPE. VOLUME=210 DEVICE_CLASS=TAPE
```

Note: *You can use the CREATE command for temporary tape files instead of the MOUNT command.*

For more information on the MOUNT command, see 7.2.20

2.20.2. Moving a Tape

If you need to move to a specific file on a tape that already has other files on it, enter

```
POSITION FILE=filename.  COUNT=exp
```

Where *exp* is a positive integer (for example, 1, 2, 3). This tells IPF 1100 to move forward a number of files relative to its current position on the tape.

For example, if you are at file 1 on a tape with the name TAPE1, and want to move to file 3, enter

```
POSITION FILE=TAPE1.  COUNT=2
```

To move backward on the tape, use the words DIRECTION=BACKWARD on the POSITION command. For example, if you are at file 3 on this tape and want to move to file 2, enter

```
POSITION FILE=TAPE1.  COUNT=1 DIRECTION=BACKWARD
```

To rewind a tape to its beginning, specify a value of 0 for COUNT. For example:

```
POSITION FILE=TAPE1. COUNT=0.
```

For more information on the POSITION command, see 7.2.26.

2.20.3. Copying Data to and from Tapes

To copy data to and from tapes, use the COPY command and specify the names of the mass-storage files or tape files in the FROM= and TO= fields. (The mass-storage file must be a data file or a directory.) For example, to copy file A*B to tape TAPE1, enter

```
COPY FROM=A*B.  TO=TAPE1.
```

To copy tape BACKUP to file NEW*FILE, enter

```
COPY FROM=BACKUP.  TO=NEW*FILE.
```

For tape-to-tape copies, first make sure you mount and position both tapes. Then enter a COPY command specifying COUNT= followed by the number of files you want to copy. For example, to copy five files from tape TAPE1 to tape TAPE2, enter

```
COPY FROM=TAPE1.  TO=TAPE2.  COUNT=5
```

Note: *You can copy only data files or entire directories to or from tapes.*

Here is an example of working with a temporary tape file:

```
▶ ~C~▶CREATE FILE=ABC.  LIFE=TEMP     ①
▶ ~C~▶MOUNT FILE=MYTAPE.  VOLUME=240 DEVICE_CLASS=TAPE     ②
▶ ~C~▶POSITION FILE=MYTAPE.  COUNT=2     ③
▶ ~C~▶COPY FROM=MYTAPE.  TO=ABC.     ④
▶ ~C~▶POSITION FILE=MYTAPE.  COUNT=0     ⑤
▶ ~C~▶COPY FROM=DEF.  TO=MYTAPE.     ⑥
▶ ~C~▶
```

Explanation

① Create a temporary file named ABC.

② Create a temporary tape file named MYTAPE, and identify it to IPF 1100 as being the magnetic tape with the external label 240.

③ Move the tape to the third file.

④ Tell IPF 1100 to copy the contents of the third file on the tape into your temporary file ABC.

⑤ Rewind the tape.

⑥ Copy the data file DEF. onto the tape.

2.21. Cataloging Tape Files

Instead of using a tape file known only to your system while the tape is actually mounted, you can catalog it.

One reason to catalog a tape file might be if you use tape files for which you specify many keyword parameters, or have multireel tape files. By cataloging a tape file, you specify all the processing information only once on a CREATE command. On subsequent use of the cataloged tape file, you only have to enter a short ATTACH command.

When you catalog a tape file using the CREATE command, include the same type of information you specify using the MOUNT command for temporary tape files. At a minimum, you must enter

```
CREATE FILE=filename. DEVICE_CLASS=TAPE
```

The *filename* must be a data file, not a directory name. The system operator will assign you a blank tape and give you its tape number.

To use the cataloged tape file, enter an ATTACH command

```
ATTACH FILE=filename.
```

If you want to use one of your own tapes, you can also specify VOLUME= *vol* on the CREATE command. For example:

```
CREATE FILE=SAVE*TAPE. DEVICE_CLASS=TAPE VOLUME=2345
```

2.22. Releasing a Tape File

If you are finished using a tape file but want to retain the tape drive for further use, enter the following command:

```
FREE FILE=filename. ACTION=RETAIN_DRIVE
```

IPF 1100 frees your tape file, but not the tape drive assigned to you. For example, if you enter the following command, the system operator removes your tape from the tape drive, but you can still use the tape drive for other tape files:

```
FREE FILE=MY*TAPE.  ACTION=RETAIN_DRIVE
```

Section 3
The Workspace and the Lookspace

This section shows how to tell IPF 1100 which file you want to work on and how to save the changes that you make to it. You use EDIT 1100 to make changes to a copy of your file in your workspace. The *IPF 1100 EDIT 1100 User's Guide* explains the actual text editing commands.

3.1. Using the Workspace and the Lookspace

The workspace is used to update the contents of an existing file, or to create and enter images into a new file. The lookspace, which is independent of the workspace, allows IPF 1100 to access files more quickly. The lookspace is used only to browse through a file without making any changes; you cannot update files using the lookspace. The lookspace has certain advantages in some circumstances that are described later in this section.

The OLD command copies a symbolic file into the workspace or lookspace. You can specify whether you want to copy the file into the workspace or lookspace by specifying your choice after a comma after the filename. For example, to copy the file into the workspace where you can update it, enter

```
OLD MYFILE.ELT,WORKSPACE
```

To copy the file into the lookspace where you can scan it (but not change it), enter

```
OLD PRINTFILE.,LOOKSPACE
```

You can move from one area to the other by using the SWITCH command. When you want to edit the other area, use the SWITCH command, and specify which area you want to edit as follows:

```
SWITCH LOOKSPACE
SWITCH WORKSPACE
```

The system variable $SWITCH is set to whatever area is currently being edited, WORKSPACE or LOOKSPACE.

When you enter editing commands, they are applied to whichever editing area is currently "active." While you are browsing through the lookspace, only those commands that do not change the lookspace are allowed. While you are editing the workspace, update commands are allowed.

When you are editing in one area, and you specify the other area on your next OLD command, that area becomes available for editing. IPF 1100 automatically sets $SWITCH to the area you specified on the OLD command.

When one editing area is active, the next OLD command with no area specified copies the specified symbolic file into the active area. In other words, the OLD command with no area specified uses the value of $SWITCH to determine where to copy the symbolic file.

Here are some examples:

To copy FILE.ELT into the workspace so you can make updates, enter

```
OLD FILE.ELT,WORKSPACE
```

To copy the print file into the lookspace and scan for errors, enter

```
OLD COMPILER*OUTPUT.,LOOKSPACE
```

To go back to the workspace and make more updates, enter

```
SWITCH WORKSPACE
```

To replace FILE.ELT, enter

```
REPLACE
```

To copy data file FILE2. into the workspace and make updates, enter

```
OLD FILE2.
```

To copy another print file into the lookspace, enter

```
OLD PRINTFILE.,LOOKSPACE
```

3.2. The Workspace

The workspace is the area you use to make changes to your files. IPF 1100 makes the specified changes first to a temporary area and then copies the temporary area into your file. However, IPF 1100 neither copies the file you want to work on into the workspace nor copies the workspace back into your file automatically.

IPF 1100 assigns the workspace when you call IPF 1100. The name of this file is

*project-id**IPF$*original-run-id*.

If this file does not exist, IPF 1100 creates it for you. Do not attempt to directly access this file (for example, do not specify this file on a CREATE, FREE, or PURGE command).

Depending on the IPF 1100 configuration option used at your site, the workspace file either is

- Always re-created when you call IPF 1100 if your previous session did not terminate due to a system crash. This results in your workspace file always being empty when you call IPF 1100.
- Re-created only on your first call to IPF 1100 during any one OS 2200 session (i.e., from signing on to your system to entering a @FIN command). This results in your workspace containing whatever it contained when you last exited (using a LOGOFF command) IPF 1100 during the same OS 2200 session.

- Re-created only when the file does not already exist. IPF 1100 never deletes your workspace for you. It remains on your system for the normal lifetime of a permanent file. This results in IPF 1100 initializing your workspace only when it must create it.

You use EDIT 1100 to make changes to an existing file or insert images into a new file (see the *IPF 1100 EDIT 1100 User's Guide*). You call EDIT 1100 by entering EDIT 1100 commands.

1. Copy file OLD.FILE contents into workspace (using OLD command).
2. Make changes to workspace (using EDIT 1100).
3. Copy workspace contents back to file OLD.FILE (using a REPLACE command).

You can either update the contents of an existing file using the IPF 1100 OLD command or create a new file using the IPF 1100 NEW command. In either case, when you are finished working with the file in your workspace, you must save it.

You must enter an OLD command to copy the file into your workspace, and afterwards a SAVE or REPLACE command to copy the temporary area into your file. We'll look at these commands in more detail later.

If you used the OLD command, use the REPLACE command to copy the contents of the workspace into a file, replacing the previous contents of that file. If you used the NEW command, you now use the SAVE command to save the workspace into a file.

Note: *The copy of the file that IPF 1100 uses in your workspace is in ASCII format.If you specify a file created in any other format, IPF 1100 changes it to ASCII when it copies the workspace back to your file.*

If another user who has the same run-id as you is currently using IPF 1100 when you enter @IPF, IPF 1100 uses a temporary file for your workspace instead of a permanent file after it displays these messages:

```
**IPF580 Your workspace file is exclusively attached to another run.
**IPF230 A temporary file will be used for your workspace file.
```

This temporary workspace file exists for the duration of your OS 2200 session (until you enter an @FIN command). Until that time, IPF 1100 reuses this temporary workspace file in subsequent IPF 1100 sessions if your site does not have the first configuration option previously listed.

In the NEW, OLD, SAVE, REPLACE, and NAME command examples, you can specify any of the following for *filename*:

- No file (for the SAVE and REPLACE commands), and IPF 1100 assumes the name found in the system variable $FILENAME
- Just a file (for example, XYZ), and IPF 1100 assumes your current work directory as the corresponding directory

- A fully qualified filename (such as, *directory.file*) (for example, ABC*DEF.GH)

If you are using data files (for example, ABC*DATA.), you cannot use the SAVE command.

3.3. Creating a New File in the Workspace

To create a new file in your workspace, use the NEW command. The format of this command is

```
NEW FILE=filename
```

where *filename* is the new name of the workspace (and the name under which you probably will want IPF 1100 to save the new file later). After you enter a NEW command, IPF 1100 erases the contents of the workspace and gives it a new name.

After entering a NEW command, you can use EDIT 1100 commands to insert data into the workspace and manipulate it.

The following screen shows some examples of the NEW command follow.

```
▸ ~C~▸SET $DISPLAY:=NUMBER
▸ ~C~▸NEW FILE=EXAMPLE      ①
▸ ~C~▸NEW FILE=MDR*WORK.A     ②
▸ ~C~▸NEW     ③
▸**IPF1230 You forgot to enter the FILE keyword parameter.
▸NEW FILE=
▸ ~C~▸NEW FILE=KQT*SEC5.B     ④
▸ ~C~▸NUMBER     ⑤
▸0010.00▸GREEN     ⑥
▸0020.00▸BLUE
▸0030.00▸BLACK
▸0040.00▸YELLOW
▸0050.00▸*END     ⑦
▸ ~C~▸     ⑧
```

Explanation

① After telling IPF 1100 you want line numbers displayed before each image, erase the contents of your workspace and name it EXAMPLE. IPF 1100 assumes the directory name is that of your work directory.

② Erase and change the name of your workspace again, this time to MDR*WORK.A. Specify an existing directory (MDR*WORK.), instead of letting IPF 1100 default to your work directory.

③ Enter a NEW command without specifying a filename. IPF 1100 displays an error message.

④ You enter an acceptable NEW command including a filename (KQT*SEC5.B).

⑤ Enter the EDIT 1100 command NUMBER, which lets you enter data into your workspace one line at a time. (do not confuse this with the system variable $DISPLAY, which you set to NUMBER to display line numbers [10, 20, and so forth.]).

⑥ IPF 1100 assigns a line number to each image you enter into your workspace, beginning at 10 and increasing the line number each time by 10.

⑦ Enter the command *END. EDIT 1100 stops the automatic line numbering and input solicitation.

⑧ IPF 1100 returns the cursor, waiting for your next command.

The example above and subsequent examples assume you have set the system variable $DISPLAY to NUMBER (via SET $DISPLAY:=NUMBER) to display the line numbers.

System variables control and help define the IPF 1100 environment. They all begin with a dollar sign ($). Many of them, such as $DISPLAY, are like switches. Enter SET $DISPLAY:=NUMBER to display line numbers before each image. Enter SET $DISPLAY:=NONUMBER to avoid displaying line numbers before each image. See Section 5 for more information on the system variables for Command Language.

If $DISPLAY is set to NONUMBER, then entering a NUMBER command results in IPF 1100 displaying a prompt of ▶▶ instead of ▶number▶. see the *IPF 1100 EDIT 1100 User's Guide* for more information on the NUMBER command.

For more information on the NEW command, see 7.2.22.

3.4. Saving a New File from the Workspace

After you create a new file in the workspace (using a NEW command) and work in it (using EDIT 1100 commands), use the SAVE command to copy the data in the workspace into a file. Its format is

```
SAVE FILE=filename
```

where *filename* is the name of the file where IPF 1100 should save the workspace.

If *filename* already exists in the directory, IPF 1100 displays an error message. If you do not specify *filename*, IPF 1100 uses the current name of your workspace (supplied on a previous NEW or NAME command). If you previously entered a NEW or NAME command without specifying a directory name and now enter a SAVE command without a directory name, IPF 1100 saves the workspace to your work directory.

Note: *If you have an IPF 1100 procedure created prior to level 3R2 in an omnibus file, with the same name as the symbolic file you want to save, you must use REPLACE. (See 3.6).*

Using a SAVE command does not change the name or contents of the workspace. Here are some examples of the SAVE command:

```
▸ ~C~▸CREATE FILE=JANDJ.      ①
▸ ~C~▸SET $DISPLAY:=NUMBER     ②
▸ ~C~▸NEW FILE=JANDJ.RHYME     ③
▸ ~C~▸NUMBER      ④
▸0010.00▸Jack and Jill went up a hill to fetch      ⑤
▸0020.00▸a pail of water; Jack fell down
▸0030.00▸and broke his crown, and Jill
▸0040.00▸came tumbling after.
▸0050.00▸*END
▸ ~C~▸SAVE      ⑥
▸ ~C~▸SAVE   MG*BACKUP.RHYME      ⑦
▸ ~C~▸
```

Explanation

① Create a new directory JANDJ.

② Tell IPF 1100 to display line numbers.

③ Enter a NEW command, erasing the workspace and renaming it JANDJ.RHYME.

④ Enter an EDIT 1100 NUMBER command, which solicits lines of input, and assigns line numbers to them starting at 0010.00 and incrementing each one by 10.

⑤ Enter four lines of input, then a line with *END to terminate the automatic line numbering and input solicitation.

⑥ Enter a SAVE command without specifying a filename. IPF 1100 saves the contents of the workspace to a file named RHYME of directory JANDJ.

⑦ Enter another SAVE command, this time specifying a filename of MG*BACKUP.RHYME. IPF 1100 saves another copy of the workspace to a file named RHYME of directory MG*BACKUP.

Note: *Entering DISPLAY $FILENAME will always show you the current name of your workspace.*

Remember, on new files included in directories, use a SAVE command. On data files, use a REPLACE command (see 3.6).

For more information on the SAVE command, see 7.2.31.

3.5. Retrieving an Existing File

To update the contents of a file that already exists, enter an OLD command. The following format:

```
OLD FILE=filename OBJECT=WORKSPACE
```

where *filename* is the name of the file whose contents IPF 1100 copies into the workspace (thus replacing the previous contents of the workspace).

If *filename* does not exist or is not available, IPF 1100 rejects the OLD command and displays an error message. If *filename* exists, IPF 1100 discards the current contents of the workspace and copies the file with the name *filename* into the workspace. IPF 1100 changes the name and subtype of the workspace to match the new file. The subtype, also called the language type, tells IPF 1100 what kind of data is in a file. For example, it might be a FORTRAN program or an IPF 1100 procedure. see Section 4 for more information on running programs.

The *filename* can be a file within a directory or it can be a data file.

Here is an example using an OLD command:

```
▸ ~C~▸SET $DISPLAY:=NUMBER     ①
▸ ~C~▸OLD FILE=COLORS OBJECT=WORKSPACE      ②
▸ ~C~▸PRINT ALL      ③
▸0010.00 Red
▸0020.00 White
▸0030.00 Black
▸0040.00 Yellow
▸0050.00 Light Blue
▸0060.00 Brown
▸0070.00 Grey
▸0080.00 Orange
▸0090.00 Light Green
▸ ~C~▸CHANGE STRING=/Light/Dark/ RANGE=ALL REPEAT=ALL      ④
▸0050.00 Dark Blue      ⑤
▸0090.00 Dark Green
▸ ~C~▸
```

Explanation

① Tell IPF 1100 to display line numbers.

② Enter an OLD command, telling IPF 1100 to copy into the workspace the file named COLORS from your work directory.

③ Print the nine images in your workspace.

④ Enter an EDIT 1100 CHANGE command, changing all occurrences of the string "Light" in your workspace to the string "Dark".

⑤ EDIT 1100 changes the string "Light" to "Dark" in lines 50 and 90 and displays them. At this point, IPF 1100 has changed these lines only in your workspace, not in the file COLORS itself. (To change the file itself, enter a REPLACE command.)

Notes:

- *Many of the examples in this manual use an OLD command to retrieve the images in a file. To duplicate these examples: enter a NEW command instead of the OLD command, enter the images shown after the PRINT ALL command, then enter a SAVE command.*
- *IPF 1100 procedures created prior to level 3R2 were stored in omnibus files with a subtype of IPF that identified them as IPF 1100 procedures. These omnibus files can also be retrieved by the OLD command and copied into the workspace. If you have both a pre-level 3R2 procedure file and a symbolic file with the same name, the procedure file will always be retrieved if you give that filename on an OLD command.*

To examine a file without updating it, copy the file into your lookspace:

```
OLD FILE=filename OBJECT=LOOKSPACE
```

For more information on the OLD command, see 7.2.23.

3.6. Saving an Old File from the Workspace

To save the data in your workspace, copy it to an existing file by using a REPLACE command. Its format is

```
REPLACE FILE=filename
```

where *filename* is the name of the file into which you want to copy the workspace. If you do not specify *filename*, IPF 1100 uses the current name of the workspace (i.e., whatever name you last specified on the most recent OLD or NAME command). The *filename* can be a file within a directory or it can be a data file. You will receive an error message if *filename* does not exist.

Here is an example using the REPLACE command (continuing with the same example as shown in the previous subsection):

```
▸0050.00 Dark Blue
▸0090.00 Dark Green
▸ ~C~▸OLD FILE=OCEANS      ①
▸ ~C~▸PRINT ALL      ②
▸0100.00 Atlantic
▸0110.00 Pacific
▸0120.00 Indian
▸ ~C~▸CHANGE /Atlantic/Arctic/,ALL      ③
▸0100.00 Arctic      ④
▸ ~C~▸REPLACE      ⑤
▸ ~C~▸OLD FILE=COLORS      ⑥
▸ ~C~▸PRINT ALL      ⑦
▸0010.00 Red
▸0020.00 White
▸0030.00 Black
▸0040.00 Yellow
▸0050.00 Light Blue
▸0060.00 Brown
▸0070.00 Grey
▸0080.00 Orange
▸0090.00 Light Green
▸ ~C~▸
```

Explanation

① Enter another OLD command, telling IPF 1100 to copy into the workspace the file OCEANS from your work directory. IPF 1100 defaults to the current value of $SWITCH which is originally the workspace. Realize that IPF 1100 first erases the contents of the workspace, and then copies file OCEANS into it. Thus, the previous changes you made to the workspace for file COLORS are lost! You will see this at ⑦.

② Display the three images in your workspace.

③ Tell IPF 1100 to change the first occurrence of the string "Atlantic" to "Arctic".

④ EDIT 1100 changes line 100.

⑤ Enter a REPLACE command. IPF 1100 copies the workspace back to the file OCEANS.

⑥ Enter another OLD command, telling IPF 1100 to again copy the file COLORS back into the workspace.

⑦ Tell IPF 1100 to display the contents of the workspace. You can see that IPF 1100 never saved the previous changes you made to the workspace (shown in the preceding example) because you did not enter a REPLACE command before entering your first OLD command (at ①).

On new files within directories, use a SAVE command, but on data files, always use a REPLACE command.

Note: *If you have an IPF 1100 procedure created prior to level 3R2, in an omnibus file, with the same name as the symbolic file you want to copy the workspace into, you must use REPLACE. The pre-level 3R2 procedure can be retrieved with the OLD command, so IPF 1100 considers this an existing file, which means only REPLACE can be used. Only symbolic files can be created by SAVE and REPLACE. Any IPF 1100 procedure created before level 3R2 will be purged when REPLACE is used to copy the workspace into a symbolic file with that name. The new symbolic file can be executed as an IPF 1100 procedure as long as it still has a subtype of IPF.*

For more information on the REPLACE command, see 7.2.28.

3.7. Changing the Name of the Workspace

To change the name of your workspace without changing its contents, enter a NAME command:

```
NAME NEW_NAME=filename
```

where *filename* is the new name for the workspace. IPF 1100 leaves the contents of the workspace unchanged. You may want to use this command if you enter an incorrect filename. For example:

```
▶ ~C~▶NEW FILE=ABC
▶ ~C~▶NAMENEW_NAME=XYZ
▶ ~C~▶DISPLAY $FILENAME
▶XYZ
▶ ~C~▶
```

This example changes the name of the workspace from ABC to XYZ.

When you initially call IPF 1100 (via @IPF), the workspace is named NAME$. After entering data, use the NAME command to change its name when you want to save or replace it.

For more information on the NAME command, see 7.2.21.

3.8. Changing the Subtype of the Workspace

Use the TYPE command to set or change the subtype of the workspace. If you are writing programs or IPF 1100 procedures, you need to use this command. If you are not writing your own programs or IPF 1100 procedures, you can skip this section.

Each file has a type and a subtype associated with it. The type tells IPF 1100 which kind of file it is: symbolic, relocatable, absolute, etc. (More about this in Section 4.) The subtype tells IPF 1100 what kind of data is in the file: a FORTRAN program, an IPF 1100 procedure, and so forth.

The format of the TYPE command is

```
TYPE LANGUAGE=subtype
```

where *subtype* is the desired language subtype of the workspace. See 7.2.34 for more information about subtypes and the TYPE command.

Use a TYPE command before saving and executing a new program you have created. IPF 1100 saves the language subtype of the program when you save or replace the program.

The following example changes the file to language subtype FOR (for a FORTRAN program).

```
TYPE LANGUAGE=FOR
```

The following is an example of using a TYPE command. (It assumes the system variable $DISPLAY is set to NUMBER.)

```
▸ ~C~▸NEW FILE=SHORT      ①
▸ ~C~▸NUMBER      ②
▸0010.00▸ 10 FORMAT('This is a short FORTRAN program')      ③
▸0020.00▸    WRITE (6,10)
▸0030.00▸    END
▸0040.00▸*END      ④
▸ ~C~▸TYPE LANGUAGE=FOR      ⑤
▸ ~C~▸SAVE      ⑥
▸ ~C~▸LIST      ⑦
▸-------------------------------------------------------------------------------
▸DIRECTORY OF RUNID*USERID.
▸-------------------------------------------------------------------------------
▸NAME TYPE SUBTYPE
▸---- ---- ------
▸SHORTSYM    FOR
```

Explanation

① Enter a NEW command, erasing and renaming the workspace SHORT.

② Enter a NUMBER command. IPF 1100 solicits lines of input, automatically assigning line numbers to them, beginning at 10, and increasing each line number by 10.

③ Enter a 3-line FORTRAN program, ending with END.

④ End the automatic line numbering and solicitation with *END. The *END command on line 30 is a FORTRAN statement that tells the FORTRAN compiler that it is at the end of the program.

⑤ Enter a TYPE command telling IPF 1100 to mark the language subtype of the workspace as FOR.

⑥ Save the program.

⑦ Enter a LIST command to verify that IPF 1100 has saved the file as type FOR. When you subsequently run your program, IPF 1100 knows the program's language type is FOR.

When you create IPF 1100 procedures, specify a value of IPF for type (see the *IPF 1100 Procedures User's Guide*).

3.9. The Lookspace

The lookspace is the area where you can only browse (or look) at a file. To copy the file into your lookspace, use the command OLD FILE=*filename* OBJECT=LOOKSPACE. You can use only these editing commands in the lookspace.

- PRINT
- GO
- LOCATE
- MERGE (where you are merging from the lookspace, not into it)
- BOTTOM
- SITE
- TOP

Any editing commands that change data images in the lookspace are not allowed (for example CHANGE, RENUMBER, MOVE, and DELETE).

The lookspace is independent of the workspace and exists only for the duration of your IPF 1100 session. It is always created as a temporary file. The current content of the lookspace or workspace is not affected when editing within the other.

Because the lookspace is read-only, IPF 1100 can access the more file quickly. Larger files, like print files or compiler listings, can be scanned for errors more quickly in the lookspace than in the workspace.

There are two restrictions on using the lookspace:

- It can only be accessed in a top-to-bottom order. This means that the beginning line numbers specified for the RANGE keyword parameter on the PRINT or LOCATE command must be smaller than the ending line number specified. For example, PRINT 10:200.
- Certain commands that affect the lookspace or change certain characteristics of the lookspace are not allowed. These include the SAVE, REPLACE, NEW, NAME, and TYPE commands.

If you enter these commands while using the lookspace, IPF 1100 displays an error message explaining that you cannot perform that action to the lookspace.

Section 4
Running Programs

This section shows how to run user programs, call system processors (such as language compilers), call IPF 1100 procedures, and execute commands.

If you are not writing programs, go to 4.3.

4.1. Introduction

After you create a program in your workspace (using the editing commands of EDIT 1100), you will want to run it (also called executing it). You can do this in one of two ways:

- Enter the RUN command (just once)
- Enter a couple of "Call" commands, followed by an XQT command.

Here are the steps to create and run a new program:

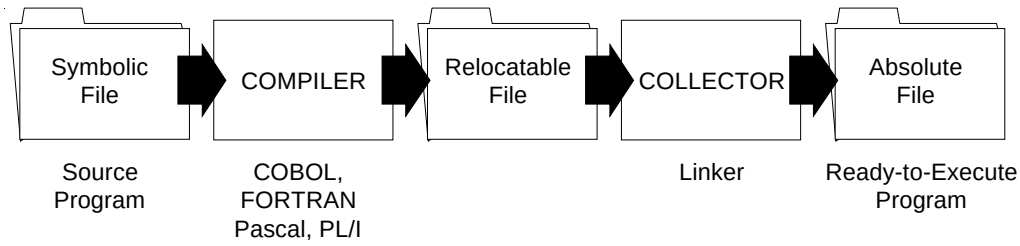
1. Use a NEW command to erase the workspace and assign it a new name.
2. Enter your program in your workspace (using EDIT 1100).
3. Use a TYPE command to tell IPF 1100 what language subtype the program is.
4. Enter a SAVE command (if you want to use this program in future sessions).
5. Enter a RUN command.

4.2. Running User Programs

When you want to execute a program, enter a RUN command. The program that you want to execute must be either in your workspace (following an OLD or NEW command), or in a file within a directory (the result of a previous SAVE or REPLACE command).

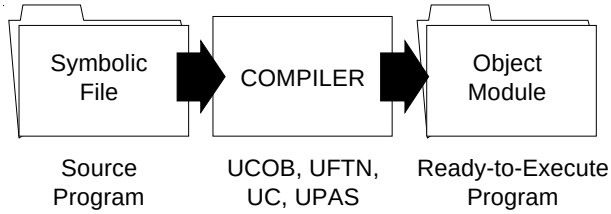
Each file within a directory has a type associated with it. This type appears when you use a LIST command (see 2.10). Types include:

Symbolic	Your program as you wrote it in a higher level language (for example, FORTRAN, COBOL, PL/I, Pascal) or assembly language. Before you can run this program, you must first process it through a compiler, which translates your English-like commands into a machine language your computer understands.
Relocatable	The output from a compiler (or assembler). This is an intermediate form of your program. This relocatable is not ready to run yet. It must go through the collector, which links it together with any other programs, subroutines, and so on, that it may need to generate an absolute. Often relocatables call on standard libraries of routines kept in a special file on your system. Having this intermediate step (of relocatables) encourages modular programming. You might have one large program that comprises many smaller ones. If you want to make some changes to just one of the smaller ones, only recompile the one small program. (Then collect all the programs together.)
Absolute	The collector takes one or more relocatables and produces an absolute. An absolute is a program that is in ready-to-execute form (it has been collected). The following figure shows how an absolute is produced.



Object module

A new kind of file within a directory. An object module replaces the relocatable and absolute files. Object module files are ready for execution and need not be explicitly collected in one absolute file. They are created by using specific object module producing compilers (for example, the Universal Compiling System). The following figure shows how an object module is produced.



When you enter a RUN command, IPF 1100 compiles, collects, and executes your source program. However, it does not save relocatables, absolutes, or object modules. If you want to save them, use the "Call" command (see 4.3).

Before you enter a RUN command, you must also identify the language subtype of your program (using a TYPE command; see 7.2.34).

Also, compilers often generate a "call" to a subroutine stored in a relocatable library. This relocatable library contains many standard sets of instructions that programs use over and over again.

If you want to use one of your own directories as a library, you must prepare it with the LIBRARY command. This allows the Collector to search the library for the subroutines that are needed by your program. If you have caused a new relocatable to be placed in a library, you must use the LIBRARY command on that directory again before it can be reused as a library.

A library is arranged in a way that makes it easier to find the required subroutines, during the course of executing your program. The library has a table that holds a list of the different subroutines available in the library. This table is built through the LIBRARY command. Its format is

```
LIBRARY FILE=directory-name-list.
```

The directory name is the name of the directory to be prepared as a library. If you specify a data file or an individual filename, then an error message appears. If you do not specify a directory name, IPF 1100 assumes the work directory.

For more information on the LIBRARY command, see 7.2.17.

The simplest format of the RUN command is to just enter

```
RUN
```

IPF 1100 compiles the program in your workspace using the default system compiler associated with the language type of your workspace. IPF 1100 then collects (that is, links) the program using the default system library, and then executes the program.

Here is an example:

```
▶ ~C~▶NEW FILE=SHORT     ①
▶ ~C~▶10.00 10 FORMAT( 'This is a short FORTRAN program' )      ②
▶ ~C~▶20.00    WRITE (6,10)
▶ ~C~▶30.00    END
▶ ~C~▶TYPE LANGUAGE=FOR      ③
▶ ~C~▶RUN      ④
▶FTN 11R2L 09/04/94-09:23(0,)
▶
▶END FTN 5 IBANK 15 DBANK
▶Collector 33R1C (930204 1925:45) 1994 Sep 04 Wed 0923:26
▶START=004714, PROG SIZE(I/D)=2001/2079
▶END MAP. ERRORS: 0 TIME: 10:792   STORAGE: 17371/7/036177/0101177
▶This is a short FORTRAN program
▶ ~C~▶
```

Explanation

① Create a new file SHORT in your workspace.

② Enter a very small FORTRAN program.

③ Identify the language subtype as FOR (a FORTRAN program).

④ Enter a RUN command. IPF 1100 compiles the program in the workspace using the FORTRAN compiler, collects it, and executes it. (At this point, nothing is saved.)

If you need to use a compiler other than the default system compiler, you must specify it. For example:

```
RUN COMPILER=MY*UCOB.UCOB
```

Note: *Some system compilers produce an object module. If the default system compiler for your workspace or file belongs to this category, an object module is produced and executed with no collection step required. (For example, the default compiler for a Pascal program is UPAS; it produces an object module that is executed immediately upon completion of the compilation step.)*

Some language types are associated with two different default compilers, one producing an object module and one producing an absolute. In such cases, the system variable $UCSCOMPILER determines which compiler is used. (For example, a FORTRAN program compiled with $UCSCOMPILER set to TRUE will produce an object module. The same program compiled with $UCSCOMPILER set to FALSE will produce an absolute.)

If the file (containing the program) you want to execute is not in your workspace, enter

```
RUN FILE=filename
```

where *filename* is the name of a file within a directory.

The *compiler-name* must be a file within a directory. For example, to execute a FORTRAN program in the file FORMULA included in the directory CHEM:

```
RUN FILE=CHEM.FORMULA COMPILER=FTN*FTN.FTN
```

where the name of the FORTRAN compiler is FTN*FTN.FTN.

To specify the name of the file or files containing the relocatable libraries you want the RUN command to use when collecting your program, enter

```
RUN FILE=filename LIBRARY=library-name-list
```

To execute the previous FORTRAN program using the relocatable library SPECIAL*RLIB, enter

```
RUN FILE=CHEM*FORMULA. COMPILER=FTN*FTN.FTN LIBRARY=SPECIAL*RLIB.
```

For more information on the RUN command, see 7.2.30.

4.3. Calling a Procedure or System Processor

You can use the "Call" command to compile your program using standard language processors (for example, COBOL, FORTRAN, PL/I, BASIC) with standard options. Then use another "Call" command to collect your program. To execute your program, use the XQT command (see 4.5).

Note: *If you specially build your program to handle parameters (for example, make it a processor), you can use the "Call" command to execute your program.*

You can also use the "Call" command to call other processors (for example, documentation processors, utilities, and so on), as well as IPF 1100 procedures.

The simplest format of the "Call" command is

```
filename
```

where *filename* is the name of the file containing the processor or procedure you want to call.

For example, to call the procedure CALCULATE, enter

```
▸ ~C~▸CALCULATE
```

Note: *IPF 1100 Procedures has an expanded format of this "Call" command, allowing you to specify any number of keywords with any names.*

You can also use an expanded format of the "Call" command:

```
filename arg-set
```

In this expanded format, *arg-set* is the set of arguments (if any) required by the program you want to execute (or the processor you are accessing). You can specify up to 20 of these arguments by position or using keywords (F1=, F2=, F3=, and so on), depending on how the executing program expects them.

You can specify a list of options for filename by entering

```
filename arg-set OPTIONS=opts
```

For example, to use the ASCII COBOL compiler, enter one of the following:

```
ACOB SYM.PGM,REL.PGM,,1 OPTIONS=CESV
ACOBF1=SYM.PGM F2=REL.PGM F4=1 OPTIONS=CESV
```

When you enter a "Call" command, IPF 1100 executes the absolute or object module in the specified file. If there is no absolute or object module with that name, then IPF 1100 executes the IPF 1100 procedure by that name if one exists.

Use care when you use the "Call" command in the following instances:

- If the filename you specify begins with a digit (for example, 123*file.abs, 123.abs, or 123abs), IPF 1100 responds with the error message

  ```
  **IPF1370 You forgot to enter a blank after the line number
  ```

  This is because a digit at the beginning of a command tells IPF 1100 that you wish to use the implicit INSERT command of EDIT 1100 to generate a new line in the workspace.

  To avoid this problem, you can either avoid using files with digits as the leading characters, use the USE command to assign a use name to the file before entering your "Call" command, or enclose the entire command in the value of $DELIMITERCHAR (usually the double-quote character).

- If you try to specify a filename by using an implied qualifier (for example, *file.abs), IPF 1100 responds with the error message

  ```
  **IPF1950 IPF 1100 does not currently support omnipresent commands
  ```

  The system variable $OMNIPRESENTCHAR specifies the special character that IPF 1100 uses to recognize omnipresent commands. Currently, the only omnipresent command is the *END command used to terminate EDIT 1100 input mode.

  To avoid this problem, you can either specify the qualifier in the filename, use the USE command to assign a use name to the file before entering your "Call" command, enclose the entire command in the value of $DELIMITERCHAR (usually the double-quote character), or change the value of $OMNIPRESENTCHAR to a character other than the asterisk.

Some processors are written to accept input from the person executing them. A processor might ask you to enter further information to direct its execution. For example, a compiler might wait for corrections to be specified after you call it, so that the symbolic file being compiled can be changed as the compiler processes it. Certain other processors wait for you to enter a special command before they will begin processing, and others wait for a special command before they actually finish.

You can tell these processors that you are not entering any more information, and that they can begin their processing immediately, by using the END keyword parameter. For example, you can enter

```
MASM SOURCE.PROG,SOURCE.REL END=YES
```

This tells the MASM processor that it can begin compiling SOURCE.PROG, to put the relocatable file in SOURCE.REL, and to stop when it is done so that you can enter another IPF 1100 command.

Specifying END=NO sends no such information to the processor. Depending on the way the individual processor was written, it might wait for you to enter more information, or require you to enter a special command before it terminates. (You might need to look for these special characteristics in the manual that describes the individual processor.) Some processors might behave the same way whether END=YES or END=NO.

You could call a program that adds records to a database for the new employees of a company by typing in the new information as follows:

```
▸ ~C~▸PERSONNEL END=NO
▸Enter the new employee's name
▸Smith, John
▸Enter the new employee's age
▸35
.
.
.
▸Are you all done?
▸Yes
▸ ~C~▸
```

For more information on the "Call" command, see 7.2.3.

4.4. Specifying the Search Order

When you want to call a program or IPF 1100 procedure, you type its name. When you enter a filename without a directory name, IPF 1100 searches the system, home, and work directories in the order specified by the system variable $CALLORDER. When you sign on to IPF 1100, $CALLORDER is equal to (SYSTEM, HOME, WORK). This means that IPF 1100 searches the system directories first, followed by the home directory, and finally the work directory.

Here is the sequence IPF 1100 follows when you enter a program or procedure for IPF 1100 to execute and $CALLORDER is equal to (SYSTEM, HOME, WORK):

1. Looks for a command by that name.
2. Searches system directories for a program by that name.
3. Searches your home directory for a program by that name.
4. Searches your work directory for a program by that name.
5. Searches the system procedure directory for a procedure by that name.
6. Searches your home directory for a procedure by that name.
7. Searches your work directory for a procedure by that name.

Here are some examples:

STAT
Calls the procedure STAT

PROC_1
Executes your procedure PROC_1

FTN FILE=NSF.SUM
Compiles the FORTRAN program NSF.SUM

You can set $CALLORDER in several ways, depending on how and when you want your directories searched.

If $CALLORDER is equal to (HOME), the search order is for

1. A command by that name
2. A program in your home directory by that name
3. A procedure in your home directory by that name

If $CALLORDER is equal to (SYSTEM, WORK), the search order is for

1. A command by that name
2. A program in the system directories by that name
3. A program in your work directory by that name
4. A procedure in your system procedure directory by that name
5. A procedure in your work directory by that name

If $CALLORDER is equal to (WORK, SYSTEM, HOME), the search order is for

1. A command by that name
2. A program in your work directory by that name
3. A program in the system directories by that name
4. A program in your home directory by that name
5. A procedure in your work directory by that name
6. A procedure in your system procedure directory by that name
7. A procedure in your home directory by that name

The following examples assume that

- MYSYSTEM is your system procedure directory and contains a procedure called UTILITY.
- MYHOME is your home directory and contains a procedure called PROC1.
- MYWORK is your work directory and contains procedures called PROC1, PROC2, and UTILITY.

If you enter the following command and then enter PROC1, IPF 1100 executes the procedure PROC1 from the directory MYWORK.

```
$CALLORDER:=(WORK,HOME)
```

If you enter the following command and then enter UTILITY, IPF 1100 executes the procedure UTILITY from MYSYSTEM.

```
$CALLORDER:=(SYSTEM)
```

If you enter the following command and then enter PROC2, IPF 1100 tells you that it is not able to find a file called PROC2.

```
$CALLORDER:=(HOME,SYSTEM)
```

4.5. Specifying an Absolute

Use the XQT command to execute user programs when you have previously used "Call" commands to compile and collect the programs and have saved the relocatables and absolutes or object modules.

The format of the XQT command is

```
XQT FILE=filename
```

where *filename* is the name of the absolute or object module you want to execute. If you want, you can specify a list of options for filename using

```
XQT FILE=filename OPTIONS=opts
```

For example:

```
XQT FILE=GAME*EDA.GO OPTIONS=U
```

If you are finished entering input to the program being executed, you can specify this by using the END keyword parameter. Typing END=YES tells the program that you are not entering any input for it to process. The program begins executing and terminates when it is done so you can enter another IPF 1100 command.

Entering END=NO allows you to enter input to the program (if it was written to accept such input). If so, you may need to enter a special command so that the program will terminate. This also depends on the way the program was written. A program that does not expect any input will execute in the same way whether END=YES or END=NO. For example, to cause execution of a program and to tell it that you will not be entering any input, you could enter

```
XQT FILE=PAYROLL.WEEKLYEND=YES
```

Here is a sequence of steps you can use to create, save, compile, collect, and execute a program using IPF 1100. Use this alternative to the RUN command when you want to save the relocatable and absolute or object module of a program.

1. Create a program in your workspace (starting with a NEW or OLD command).
2. Save the program (SAVE or REPLACE *directory.filename*).
3. Compile the program (using the "Call" command: ACOB *directory.filename*).
4. Collect the program (if a relocatable was produced):
   a. If you use the "Call" command, enter MAP ,*directory.filename*.
   b. If you use Collector statements, enter IN *directory.filename*then enter END.
5. Execute the program (using the command: XQT *directory.filename*).

Note: *If you have specially built your program to handle parameters (that is, made it a processor), you can use the "Call" command (see 4.3) to execute your program.*

For more information on the XQT command, see 7.2.36.

4.6. Executing Commands

Use the EXECUTE command to execute a single command that you can specify as a string of characters.

The format of the EXECUTE command is

```
EXECUTE COMMAND=string-exp
```

where *string-exp* is the command you want to execute, expressed as a string. The command string can include variables and functions. (see 1.9.9.)

You can specify the control language type of the string by using the LANGUAGE=keyword parameter.

Enter EXECUTE COMMAND=*string-exp* LANGUAGE=ECL to indicate Executive Control Language (this is the default). LANGUAGE=IPF indicates that you want to execute an IPF 1100 command.

For more information on the EXECUTE command, see 7.2.13.

4.7. Interprogram Communication

You can send information to and from Exec runstreams and MASM programs using the CONDITION command to set the Run Condition Word.

4.7.1. The Run Condition Word

The Run Condition Word is a portion of memory that is assigned to each computer run. It consists of 36 pieces, or bits. Each of these bits contains a binary digit (0 or 1). When you combine 6 of these bits together you get a sixth of a word. There are 6 sixths in each word. Two of these sixths put together make up a third of a word. A half word consists of 18 bits. If you would like these concepts defined further, see the *MASM Programming Reference Manual*. IPF 1100 Command Language uses MASM naming conventions for referencing the pieces of the word. They are as follows: W for the whole word; H1 and H2 for the half words; T1, T2, and T3 for thirds; and S1 through S6 for the sixths.

The Run Condition Word contains a value that can be used in Exec runstreams or MASM programs. It is possible to send information to these programs, through the run condition word, to tell them what you would like done. These programs can then send information back, through the run condition word, to tell you what has been done.

A user program or Exec runstream must be written to retrieve the Run Condition Word, and take appropriate action based on its contents. The program or runstream can alter the Run Condition Word to affect later programs or runstreams.What part of the word is changed, and what value it can be set to, is up to the writer of the program or runstream.

4.7.2. The CONDITION Command

Use the CONDITION command to change the value of the Run Condition Word.

The format of the CONDITION command is

```
CONDITION  AREA = { T2 | T3 | S3 | S4 | S5 | S6 }
           VALUE = numeric-expression
           [ FUNCTION = { REPLACE | LAND | LOR | LXOR } ]
```

where

AREA	is that part of the Run Condition Word to be changed
VALUE	is the numeric value to be used
FUNCTION	is the logical operation to be performed on the current value of that part of the Run Condition Word and *numeric-expression*. The default for FUNCTION is REPLACE, where *numeric-expression* takes the place of the specified area of the Run Condition Word. The other choices for FUNCTION are described in 5.7.4.

To retrieve information from the Run Condition Word, use the system function $CONDITION (see 5.7.4). For more information on the CONDITION command, see 7.2.4.

Section 5
Variables and Functions

This section explains system and user variables that pertain to Command Language. It shows how to give them values, as well as how to display them. It also explains the available system functions.

5.1. Introduction

IPF 1100 has two types of variables: system and user. A variable is an entity whose value you can change (as opposed to a constant). Every IPF 1100 user's session has a unique set of system variables and user variables. Variables let you tailor your IPF 1100 session to your specific requirements.

System variables always begin with a $ (for example, $DATE). They have predefined names, which are listed in Table 5-1.

User variables always begin with a % (for example, %TRUCK). They have the names you give them when you first use them. (See the *IPF 1100 Procedures User's Guide*. Both types of variables remain in effect for your entire IPF 1100 session.

5.2. Using System Variables

Table 5-1 lists the IPF 1100 system variables that pertain to Command Language, their abbreviations (in parentheses), and their allowed and initial values. To set a system variable, enter one of the following:

```
SET system-variable:=value
system-variable:=value
```

If you try to change the value of a system variable you are not allowed to change, IPF 1100 displays an error message:

```
**VARIABLE130 You may not change the value of this system variable.
```

For the current value of a system variable, enter

```
DISPLAY VALUE=system-variable
```

Table 5-1. IPF 1100 System Variables for Command Language

System Variable	Explanation and Values	Default Value
$BREAKPOINT ($brea, $brkpt)	Tells if the current IPF 1100 session was started in breakpoint mode or not. Values are TRUE or FALSE.	Set by IPF 1100
$CALLORDER ($call) †, ‡	Specifies the order the system, home, and work directories are searched for a file to be executed by typing the file's name, but no directory name.	(system,home, work)
$CASESENSITIVE ($case) †, ‡	When this is FALSE, IPF 1100 makes all string comparisons without regard to uppercase or lowercase alphabetic characters. Values are TRUE and FALSE.	TRUE
$CCS †	If the I18N feature is active, determines the coded character set (CCS) that IPF 1100 uses to store and display data in the current editing object and to produce output to a file. It has no effect if I18N is inactive. Possible values include any CCS name or number defined and available on your system.	1) CCS of the workspace if a valid workspace is carried forward from a previous I18N-active IPF 1100 session and its CCS is available on the system. 2) US ASCII if there is no workspace file at LOGON, if the workspace file is empty at LOGON, if the workspace file is from an IPF 1100 session where I18N was inactive, or if the workspace file CCS does not correspond to a CCS available on your system.
$COLLATE ($COLL) †	If I18N is active, determines how IPF 1100 handles character comparisons, based upon the processing rules for the specified locale. It has no effect if I18N is inactive. Possible values include any locale name or number defined and available on your system.	Value of your environment variable LC_COLLATE, if defined, or a system-defined default locale.

Legend continued

† You can set the value.

‡ You can make this variable local to a procedure.

Table 5-1. IPF 1100 System Variables for Command Language (cont.)

System Variable	Explanation and Values	Default Value
$COMMENTCHAR ($ccha) †, ‡	Contains the character currently defined as the comment delimiter (that is, anything to the right of the character on a line is a comment).	@
$COMPLETIONS ($comp) †, ‡	Determines whether or not IPF 1100 displays completion notices after a command.	FALSE
$CONTINUATIONCHAR ($cont, $contchar) †, ‡	Contains the current character that indicates continuation of images from one line to another.	&
$CTYPE ($CTYP) †	If I18N is active, determines how IPF 1100 handles case conversions, based upon the processing rules for the specified locale. It has no effect if I18N is inactive. Possible values include any locale name or number defined and available on your system.	Value of your environment variable LC_CTYPE, if defined, or a system-defined default locale.
$DATE	Contains the current date as a string in the standard format mm/dd/yy.	Current date
$DELIMITERCHAR ($deli, $delimchar) †, ‡	Contains the current character that delimits strings. Any character is legal except "$" and ",".	"
$ERRORFORMAT ($erro) †, ‡	Contains the format IPF 1100 uses to display error messages. Allowed values are: ID_ONLY, which means only the error number is displayed; TEXT_ONLY, which means just the text is displayed; or COMPLETE, which means the entire message is displayed.	COMPLETE
$FILELENGTH ($fl, $flen)	Contains the number of images in your workspace.	
$FILENAME ($fn, $fnam)	Contains the name of the workspace (resulting from a NAME, NEW, or OLD command.	NAME$
$FILESTATUS ($fils)	Specifies whether the workspace has been changed since the last OLD (into the workspace), NEW, SAVE, or REPLACE command. Possible values are MODIFIED and UNMODIFIED.	UNMODIFIED
$FUNCTIONKEY1 ($f1) †, ‡	Contains the value F1. Pressing a function key containing the value F1 tells IPF 1100 to display the system variables associated with your terminal's function keys and their corresponding values.	F1

Legend continued

† You can set the value.

‡ You can make this variable local to a procedure.

Table 5-1. IPF 1100 System Variables for Command Language (cont.)

System Variable	Explanation and Values	Default Value
$FUNCTIONKEY2 ($f2)	Contains the value F2. Pressing a function key containing the value F2 tells IPF 1100 to	F2 redisplay the screen which you are currently viewing, exactly as it was when IPF 1100 first painted it.
$FUNCTIONKEY3 ($F3) †, ‡	Contains the value F3. Pressing a function key containing the value F3 tells IPF 1100 to read the entire screen.	F3
$FUNCTIONKEY4 ($F4) †, ‡	Contains the value F4. Pressing a function key with the value F4 tells IPF 1100 to display the last input entered at the terminal on the command line.	F4
$FUNCTIONKEYn ($Fn) (where n is a number from 5 - 30) †, ‡	Determines what action will be taken by IPF 1100 when the associated function key (as indicated by n) is pressed on your terminal.	Fn
$HOMEDIRECTORY ($hd, $home)	Contains the current home directory for your session. This is one of the directories searched by IPF 1100 for a processor or an IPF 1100 procedure when no directory name is specified,	The directory specified on the IPF call line. If none is then the default value of project-id *user-id is used.
$JOBID ($jobi)	Contains the job-id of the current job (this is the same as the run-id).	job-id
$LANGUAGE ($lang)	Contains the symbolic subtype of the workspace.	SYM
$LASTINPUT ($linp)	Contains the text of the last complete command executed by IPF 1100.	LOGON command
$MULTIPLECMDCHAR ($mult, $multicmdchar) †, ‡	Contains the character used to separate commands when you type multiple commands per image. Any character is legal except the current value of $DELIMCHAR.	;
$OMNIPRESENTCHAR ($omni, $omnichar) †, ‡	Contains the character used to indicate that the command is not data and can be interspersed with data.	*

Legend continued

† You can set the value.

‡ You can make this variable local to a procedure.

Table 5-1. IPF 1100 System Variables for Command Language (cont.)

System Variable	Explanation and Values	Default Value
$PAGECALL ($pcal) †, ‡	Determines how IPF 1100 will handle the output generated by programs. By setting this variable, you can tell IPF 1100 to pause before or after your program has been executed to allow you to view all of your processor output. The possible values are NEITHER, BEFORE, AFTER, and BOTH.	NEITHER
$PAGELENGTH ($plen) †, ‡	Specifies the number of lines of output IPF 1100 will generate before a page break occurs in line mode. This can be set to an integer between 1 and 255 inclusive.	22
$PAGEWIDTH ($pwid) †, ‡	Contains the maximum number of characters for each line of IPF 1100 output when $PAGING is TRUE.	80
$PAGING ($pagi) †, ‡	Determines whether IPF 1100 will generate page breaks in output which is sent to the terminal. Possible values are TRUE and FALSE.	FALSE
$RUNMODE ($runm)	Contains the type of run which was active when you entered IPF 1100.	
$SITEID ($site)	Contains the site identification for your terminal.	
$SWITCH ($swit)	Identifies what editing object is currently being used. Possible values are WORKSPACE and LOOKSPACE.	WORKSPACE

Legend continued

† You can set the value.

‡ You can make this variable local to a procedure.

Table 5-1. IPF 1100 System Variables for Command Language (cont.)

System Variable	Explanation and Values	Default Value
$TERMINALCCS ($tccs, $termccs) †	If the I18N feature is active, determines the coded character set (CCS) in which IPF 1100 allows you to enter terminal input and to view terminal output. It has no effect if I18N is inactive. Possible values include any CCS name or number defined and available on your system.	1) CCS of the workspace if a valid workspace is carried froward from a previous I18N-active IPF 1100 session and its CCS is available on the system. 2) US ASCII if there is no workspace file at LOGON, if the workspace file is empty at LOGON, if the workspace file is from an IPF 1100 session where I18N was inactive, or if the workspace file CCS does not correspond to a CCS available on your system.
$TERMINALTYPE ($term) †	Contains the type of terminal being used for screen mode operations. IPF 1100 will use this value to determine how to display data on your terminal. Possible values are any terminal types supported by DPS 1100 (for example, UTS20, UTS60, IBM3270). NOTE: If you allow IPF 1100 to automatically set this variable, the value chosen is the most generic terminal type that applies. Therefore, it may not be exactly what your terminal is.	Blanks
$TIME	Contains a string denoting the current time of day in standard format hh :mm :ss.	Current time
$UCSCOMPILER ($ucsc) †, ‡	Determines whether the RUN command selects a UCS or non-UCS compiler to act as the default compiler.	FALSE
$USERID ($uid)	Contains the user-id of the session.	user-id

Legend continued

† You can set the value.

‡ You can make this variable local to a procedure.

Table 5-1. IPF 1100 System Variables for Command Language (cont.)

System Variable	Explanation and Values	Default Value
$WAITTIME ($wait) †, ‡	Specifies the number of seconds IPF 1100 should wait before displaying output in both line and full-screen mode. This can be set to an integer value between 0 and 511 inclusive.	0
$WORKDIRECTORY ($wd, $work) †, ‡	Contains the current work directory for your session. This directory is sometimes searched automatically by IPF 1100 for files that do not specify a directory name.	Value of $HOME DIRECTORY

Legend

† You can set the value.

‡ You can make this variable local to a procedure.

5.3. Giving a Value to a Variable

Use the SET command to establish (or reestablish) a value for a system or user-defined variable. The format of the SET command is

```
SET variable:=exp
```

where the word SET is optional.

IPF 1100 evaluates *exp* (the full expression you enter) and stores the result in the variable. For more information on allowable expressions, see 1.9.9.

Here are some examples of assigning values to system and user variables:

```
SET $OMNICHAR:="+"
$DELIMCHAR:="/"
SET %PARA3:=blue
```

Use quotation marks if there is a possibility of confusion.

For more information on the SET command, see 7.2.32.

5.4. Printing the Values of Variables

Use the DISPLAY command to display the value of variables and full expressions to the terminal or the system console. The format of the DISPLAY command is

```
DISPLAY VALUE=exp
```

If you enter DISPLAY VALUE=*exp*, IPF 1100 displays only the value for the variable or full expression you specify. IPF 1100 evaluates the expression, *exp*, and displays the value of the result. For more information on allowable expressions, see 1.9.9.

Here are some examples:

```
▸ ~C~▸DISPLAY VALUE=$OMNICHAR
▸*
▸ ~C~▸SET %UVAR:=33
▸ ~C~▸DISPLAY %UVAR/3
▸11
▸ ~C~▸DISPLAY 12+25
▸37
▸ ~C~▸DISPLAY "12+25"
▸12+25
▸ ~C~▸DISPLAY "The value of $COMPLETIONS is "& $COMPLETIONS
▸The value of $COMPLETIONS is FALSE
```

Note that anything within quotation marks is a literal; IPF 1100 displays it verbatim. In the preceding examples, because the command DISPLAY "12+25" includes the literal "12+25", IPF 1100 displays 12+25. But because the command DISPLAY 12+25 includes the full expression 12+25, IPF 1100 evaluates the arithmetic expression 12+25 and displays 37.

You can include an optional keyword parameter MESSAGE_CLASS to specify the type of message to display. You can specify values of DATA, REMARK, WARNING, or ERROR.

For more information on the DISPLAY command, see 7.2.8.

5.5. Listing Variable Names

Use the REVIEW command to retrieve a list of variable names and display them with their values. To display all the user variables you have created, enter

```
REVIEW VARIABLES=USER
```

To display all the systems variables you are allowed to see, enter the following. Each variable name is displayed with its current value.

```
REVIEW VARIABLES=SYSTEM
```

Here is an example:

```
▸ ~C~▸SET %month:=June
▸ ~C~▸SET %day:=Monday
▸ ~C~▸REVIEW VARIABLES=USER
▸ %month     June
▸ %day       Monday
```

For more information on the REVIEW command, see 7.2.29.

5.6. Soliciting a User Variable Value

During the execution of an IPF 1100 procedure, you may want the procedure to temporarily stop and request that information be entered.

The ACCEPT command lets you ask the caller of the procedure (or the system operator) to enter a value, and assigns this value to a variable. It has the format:

```
ACCEPT  VARIABLE=var
        [ PROMPT=string-exp ]
        [ FROM= { USER | OPERATOR } ]
```

The PROMPT=*string-exp* keyword parameter is optional. If you specify *string-exp*, IPF 1100 displays it as a prompt.

The FROM={ USER | OPERATOR } keyword parameter is also optional. If you specify OPERATOR, IPF 1100 displays the prompt string on the system console and requests a value from the system operator. Otherwise, the prompt appears on the user's terminal with a request for a value.

Here are some examples of ACCEPT commands:

```
ACCEPT VARIABLE=%NAME
ACCEPT %X
ACCEPT VARIABLE=%DIG PROMPT="Please type a 1-digit integer"
ACCEPT %SYSTEM, "WHAT SYSTEM IS THIS?" FROM=OPERATOR
```

For more information on the ACCEPT command, see 7.2.1 and the *IPF 1100 Procedures User's Guide*.

5.7. Specifying System Functions

A system function is a stored (or canned) operation built into IPF 1100. You call a function by typing its name (always beginning with $) and passing zero or more values (called arguments) within parentheses and separated from each other by commas. IPF 1100 returns a value in place of your function call. Often people use functions within expressions.

IPF 1100 allows function calls wherever it allows you to specify a user or system variable. The value IPF 1100 returns depends on the function you call.

Generally, IPF 1100 functions have this format:

```
$function-name (arg-1 [,arg-2 ...] )
```

where *function-name* is the name of the function you are calling (for example, $LOWERCASE), and *arg-1* to *arg-n* are arguments you are supplying as input to the function (for example, AbCdE). Thus, $LOWERCASE(AbCdE) is a correct call to the $LOWERCASE function. IPF 1100 returns a value in place of the function call (in this example, abcde).

The opening parenthesis must immediately follow the function name. If you use more than one argument, separate them with commas.

If you use strings with spaces or special characters, you must enclose them with the delimiter character (initially ").

Similar to command and keyword parameter names, you can abbreviate function names to four characters.

5.7.1. Working with Strings

The following system functions allow you to manipulate characteristics.

Finding the Length of a String ($LENGTH)

The system function $LENGTH(*string-exp*) returns the length of *string-exp* as an integer. Remember to enclose *string-exp* within delimiter characters (the current value of the system variable $DELIMCHAR) if it contains spaces or special characters.

```
▸ ~C~▸DISPLAY $LENGTH(relationship)
▸12
▸ ~C~▸DISPLAY $LENGTH("Here are four words")
▸19
▸ ~C~▸
```

Finding the Position of a Substring in a String ($SEARCH)

The system function $SEARCH(*string1*, *string2*[,*start*]) returns an integer indicating the first character number (from the left) within *string1* where it finds *string2*, starting at *start*. For example, within the word BASEBALL, the word BALL starts at character number 5.

If you do not specify *start*, IPF 1100 assumes 1. If IPF 1100 does not find *string2*, or the value you specify for *start* is less than 1, it returns a zero. If the value you specify for start is greater than the length of the string, IPF 1100 returns a zero.

If the I18N feature is active for IPF 1100 on your system and the system variable $CASESENSITIVE is set to FALSE, the $SEARCH function performs a case-insensitive search for *string2*. *string1* and *string2* will be converted to uppercase characters using the case conversion processing rules defined for the locale stored in the system variable $CTYPE (see 5.2). If I18N is inactive and $CASESENSITIVE is set to FALSE, the $SEARCH function performs a case-insensitive search for *string2* using the case conversion processing rules of the US ASCII coded character set.

```
▸ ~C~▸DISPLAY $SEARCH(handicap,cap)
▸6
▸ ~C~▸DISPLAY $SEARCH("New York, New York",York,9)
▸15
▸ ~C~▸DISPLAY $SEARCH(fish,tuna)
▸0
▸ ~C~▸SET %NAME:=JOHNSON
▸ ~C~▸DISPLAY $SEARCH(%NAME,SON)
▸5
▸ ~C~▸
```

Returning a Substring within a String ($SUBSTRING)

Whereas the system function $SEARCH returns the column number of a substring within a string, the system function $SUBSTRING (*string*, *offset* [,*length*]) returns the substring itself within *string*, beginning at column number *offset*, and *length* characters long. If you specify an *offset* that is less than 1, IPF 1100 assumes 1. If *offset* is greater than the length of *string*, IPF 1100 returns a null string.

If you do not specify *length*, IPF 1100 uses the length of *string* minus *offset* + 1. If the *length*you specify is less than 1, IPF 1100 assumes a length of zero and returns a null string.

```
▸ ~C~▸DISPLAY $SUBSTRING(BASEBALL,5,4)
▸BALL
▸ ~C~▸DISPLAY $SUBSTRING(entertainment,1,5)
▸center
▸ ~C~▸
```

Converting a String to All Lowercase Characters ($LOWERCASE)

The system function $LOWERCASE(*string-expression*) converts a string expression to a string of all lowercase characters.

If the I18N feature is active for IPF 1100 on your system, then IPF 1100 uses the case conversion processing rules defined for the locale stored in the system variable $CTYPE (see 5.2). If I18N is inactive, IPF 1100 uses US ASCII case conversion rules.

```
▸ ~C~▸DISPLAY $LOWERCASE(Cat)
▸cat
▸ ~C~▸DISPLAY $LOWERCASE(A & B & C)
▸abc
▸ ~C~▸DISPLAY $LOWERCASE("2 PEOPLE")
▸2 people
```

Converting a String to All Uppercase Characters ($UPPERCASE)

The system function $UPPERCASE(*string-expression*) converts a string expression to a string of all uppercase characters.

If the I18N feature is active for IPF 1100 on your system, then IPF 1100 uses the case conversion processing rules defined for the locale stored in the system variable $CTYPE (see 5.2). If I18N is inactive, IPF 1100 uses US ASCII case conversion rules.

```
▸ ~C~▸DISPLAY $UPPERCASE(Dog)
▸DOG
▸ ~C~▸DISPLAY $UPPERCASE(h & o & u & s & e)
▸HOUSE
▸ ~C~▸DISPLAY $UPPERCASE("2 PeOpLe")
▸2 PEOPLE
```

Removing Blanks from a String ($TRIM)

The system function $TRIM(*string-expression* [, *side*, *character*]) removes characters from the specified *side* of a string, and *side* may have the value RIGHT, LEFT, BOTH, or ALL. *Character* can be any single character.

If you do not specify *side*, IPF 1100 assumes RIGHT.

If you do not specify a character, IPF 1100 assumes a blank.

If the I18N feature is active for IPF 1100 on your system and the system variable $CASESENSITIVE is set to FALSE, the $TRIM function performs a case-insensitive removal of the character. *string-expression*and *character* will be converted to uppercase characters using the case conversion processing rules defined for the locale stored in the system variable $CTYPE (see 5.2). If I18N is inactive and $CASESENSITIVE is set to FALSE, the $TRIM function performs a case-insensitive removal of the character using the case conversion processing rules of the US ASCII coded character set.

```
▶ ~C~▶DISPLAY $TRIM("    abc    ")
▶   abc
▶ ~C~▶DISPLAY $TRIM("    abc    ",RIGHT)
▶   abc
▶ ~C~▶DISPLAY $TRIM("    abc   ",LEFT)
▶abc
▶ ~C~▶DISPLAY $TRIM("    abc   ",BOTH)
▶abc
▶ ~C~▶DISPLAY $TRIM("  a  b  c  ",ALL)
▶abc
```

Adding a "pad" to a String ($PAD)

The system function $PAD(*string*, *length* [, *side*, *character*]) changes a string to include characters on the indicated side of your string. *Side* may have the value RIGHT, LEFT, or BOTH. The resulting string will have *length* characters.

If you do not specify *side*, IPF 1100 assumes RIGHT. If you do not specify *character*, IPF 1100 assumes a blank.

```
▶ ~C~▶DISPLAY $PAD(abc,5)
▶abc
▶ ~C~▶DISPLAY $PAD(abc,5,left)
▶  abc
▶ ~C~▶DISPLAY $PAD(abc,6,BOTH,X)
▶xabcxx
```

Finding What Is At a Specific Line Number ($TEXT)

The system function $TEXT(*line-number*) returns the image *line-number* in your workspace. If you do not specify *line-number*, IPF 1100 assumes your current line number (the value of $C). If the *line-number* you specify does not exist, IPF 1100 returns a null string.

```
▶ ~C~▶PRINT ALL
▶0010.00 green
▶0020.00 blue
▶0030.00 red
▶0040.00 yellow
▶ ~C~▶DISPLAY $TEXT(10)
▶green
▶ ~C~▶GO 30
▶ ~C~▶DISPLAY $TEXT
▶red
▶ ~C~▶
```

5.7.2. Working with Numbers

The following system functions allow you to manipulate numeric expressions.

Finding the Absolute Value of a Number ($ABS)

The system function $ABS(*numeric-expression*) returns the absolute value of a number or numeric expression.

```
▶ ~C~▶DISPLAY $ABS(10)
▶10
▶ ~C~▶DISPLAY $ABS(-5*6)
▶30
▶ ~C~▶SET %X := -110.55
▶ ~C~▶DISPLAY $ABS(%X)
▶110.55
```

Finding the Integer Portion of a Number ($INTEGER)

The system function $INTEGER(*numeric-expression*) returns the integer portion of the *numeric-expression*.

```
▶ ~C~▶DISPLAY $INTEGER(5.4)
▶5
▶ ~C~▶DISPLAY $INTE(-5.4)
▶-5
▶ ~C~▶SET %Y := 10.5
▶ ~C~▶DISPLAY $INTE(%Y)
▶10
▶
```

Finding the Fractional Portion of a Number ($FRACTION)

The system function $FRACTION(*numeric-expression*) returns the fractional portion of the *numeric-expression*.

```
▶ ~C~▶DISPLAY $FRACTION(5.4)
▶0.4
▶ ~C~▶DISPLAY $FRAC(90.34)
▶0.34
▶ ~C~▶SET %portion := (-3.675)
▶ ~C~▶DISPLAY $FRAC(%portion)
▶-0.675
```

5.7.3. Converting Values from One Type to Another

The following system functions allow you to convert numeric values to strings, strings to numbers, and a character to its ASCII equivalent or vice versa.

Converting a Numeric Field to a String ($STRING)

The system function $STRING(*number*) converts a numeric value to a string (its character representation).

```
▶ ~C~▶DISPLAY $STRING(5)
▶5
▶ ~C~▶
```

Changing a String Expression to a Number ($NUMBER)

The system function $NUMBER(*string-expression)* converts a *string-expression* to a number.

The string can contain digits, a decimal point, and a plus or minus sign.

```
▸ ~C~▸DISPLAY $NUMBER("5.4")
▸5.4
▸ ~C~▸DISPLAY $NUMB("-20.63")
▸-20.63
```

Finding the Integer Representation of an ASCII Character ($ASCII)

The system function $ASCII(*character*) converts a single ASCII character or a string expression that evaluates to a single character to its integer representation in the ASCII character set.

```
▸ ~C~▸DISPLAY $ASCII(A)
▸65
▸ ~C~▸DISPLAY $ASCII(a)
▸97
```

Finding the ASCII Character of an Integer ($CHARACTER)

The system function $CHARACTER(*integer-expression*) converts an integer in the range from 0 to 511 to its corresponding character in the ASCII character set.

```
▸ ~C~▸DISPLAY $CHARACTER(65)
▸A
▸ ~C~▸DISPLAY $CHARACTER(97)
▸a
```

Note: *Many characters in the range 0 - 511 are not allowed on certain types of terminals. Displaying these characters will result in unpredictable results.*

5.7.4. Miscellaneous Functions

The following system functions allow you to manipulate the Run Condition Word, return a fully qualified name, and return the status of a specified field.

Retrieving Information from the Run Condition Word ($CONDITION)

Use this system function to display selected parts of the Run Condition Word. You can display a selected part directly, or display a part after having a logical function applied to it.

The first form of $CONDITION is

```
$CONDITION (area-name)
```

where *area-name* = W | H1 | H2 | T1 | T2 | T3 | S1 | S2 | S3 | S4 | S5 | S6

The second form is

```
$CONDITION (area-name, mask-value,  logical-function)
```

where *mask-value* is a positive integer value and *logical-function* = LAND | LOR | LXOR.

In the second form, the values of *area-name* and *mask-value* are operated on by *logical-function*, with the value of the function being the value of the logical operator.

Logical Functions

A set of logical functions is allowed in the CONDITION command and the $CONDITION function. These are the logical functions LAND, LOR, and LXOR (logical AND, logical OR, and logical Exclusive OR). The specified logical function will be performed on that part of the Run Condition word indicated by *area-name*, and the positive integer specified by *mask-value*. The result of the $CONDITION function will be the result of this logical operation. The functions are performed on the individual bits (1s and 0s) of these values. The values given are right justified, and then the functions are performed on the corresponding bits (bit 1 with bit 1...) and then the result provided from the following descriptions.

The AND Function

The AND returns a 1 if and only if both bits are set to 1.

bit	0	1
0	0	0
1	1	1

The OR Function

The OR returns a 1 when one or both of the bits are set to 1.

bit	0	1
0	0	1
1	1	1

The XOR Function

The XOR returns a 1 only if one or the other of the bits is set to 1, but not both. Thus it returns a 1 when 1 bit is set, exclusively.

bit	0	1
0	0	1
1	1	0

Here are some examples:

To return the value of the entire Run Condition Word, enter

```
▸ ~C~▸DISPLAY $CONDITION(w)
```

To return the result of performing the logical-function AND on the mask-value 1000 and the first half (h1) of the Run Condition Word, enter

```
▸ ~C~▸DISPLAY $CONDITION(h1,1000,AND)
```

For further information on the Run Condition Word, see the *ER Programming Reference Manual*.

Finding the Fully Qualified Filename ($FILE)

The system function $FILE(*filename*) returns the fully qualified name of a directory or data file.

```
▸ ~C~▸DISPLAY $FILE(filea.)
▸ABC*FILEA(1).
▸ ~C~▸SET=%a := "d."
▸ ~C~▸DISPLAY $FILE(%a)
▸ABC*D(1).
```

Getting Information on a Specific Field of a Form ($FIELDSTATUS)

The system function $FIELDSTATUS(*ipf-field-name*) returns the status of the specified field on the form that was read during the last SEND_FORM or SEND_MESSAGE command. Forms are used with IPF 1100 procedures. For more information on $FIELDSTATUS and related concepts, refer to the *IPF 1100 Procedures User's Guide*.

5.7.5. Using More Complex Expressions

You can do some interesting and powerful things by combining system functions in more complex expressions. Here are some examples:

```
▸ ~C~▸DISPLAY $SUBSTRING(CARDIOGRAM,$SEARCH(PROGRAMMING,GRAM)+3,4)
▸GRAM
▸ ~C~▸GO 110
▸ ~C~▸PRINT
▸0110.00 apple
▸ ~C~▸DISPLAY $LENGTH($TEXT)
▸5
▸ ~C~▸
```

Table 5-2 summarizes all the IPF 1100 system functions, the arguments they require, and the type of value returned.

Table 5-2. IPF 1100 System Functions

Function Name	Arguments	Value Returned	Example
$ABS(numeric-expression)	numeric-expression	Returns the absolute value of a number or a numeric expression.	$ABS(-10.6) returns 10.6
$ASCII(character)	character	Converts a single ASCII character to its integer representation in the ASCII character set.	$ASCII(A) returns 65
$CHARACTER (integer-expression)	integer-exp	Converts an integer in the range from 0 to 511 to its corresponding character in the ASCII character set. NOTE: Many characters in this range are not allowed on certain terminals. Displaying these characters could cause unpredictable results.	$CHARACTER(50) returns 2

continued

Table 5-2. IPF 1100 System Functions (cont.)

Function Name	Arguments	Value Returned	Example
$CONDITION(area-name[,mask-value, logical-function])	area-name - W, H1, H2, T1, T2, T3, S1, S2, S3, S4, S5, S6 mask-value - positive integer value logical-function - LAND, LOR, LXOR	Returns the result of performing the logical function (LAND, LOR, LXOR) on the mask-value (any positive integer value) and the portion of the Run Condition Word specified by the area-name (W, H1, H2, T1, T2, T3, S1, S2, S3, S4, S5, S6).	$CONDITION(T3) returns 63
$FIELDSTATUS ipf-field-name	ipf-field-name	Status of the specified field on the form that was read during the last SEND_FORM or SEND_MESSAGE command.	$FIELDSTATUS (form3_field4) returns 0
$FILE(file-name)	file-name	Fully qualified name of directory or data file.	$FILE(a.) returns abc*a(1).
$FRACTION (numeric-expression)	numeric-expression	Returns fractional portion of the numeric expression.	$FRACTION(5.4) returns 0.4
$INTEGER (numeric-expression)	numeric-expression	Returns integer portion of the numeric expression.	$INTEGER(5.4) returns 5
$LENGTH (string-expression)	string-expression	The length of s as an integer value.	$LENGTH("Dog") returns 3
$LOWERCASE (string- expression)	string-expression	Converts a string-expression to lowercase characters.	$LOWERCASE("AT") returns "at".
$NUMBER (string-expression)	string-expression	Converts a string-expression to a number.	$NUMBER("5.4") returns 5.4
$PAD(string,l[,s,c])	string, l -integer, side -RIGHT, LEFT, BOTH, ALL, c -single character	Adds specified number of characters to the specified side (RIGHT, LEFT, BOTH, ALL). If you omit side, IPF 1100 assumes the RIGHT side. Character is assumed to be blank.	$PAD(abc, 6, BOTH, x) returns "xabcxx")
$SEARCH(string-1, string-2 [,start])	string-1, string-2 -strings, start- integer	Returns the column number within string-1 of where string-2 is located, starting at column start. If you omit start, IPF 1100 assumes column 1.	$SEARCH ('understand','n',3) returns 9
$STRING(num)	num -number	Converts a numeric field to a string.	$STRING(5) returns "5"

continued

Table 5-2. IPF 1100 System Functions (cont.)

Function Name	Arguments	Value Returned	Example
$SUBSTRING(s, i [,l])	s- string i, l- integers	The substring of s of length l starting at the i th character. If you omit l, IPF 1100 assumes $LENGTH(s).	$SUBSTRING ('application',6,3) returns "cat"
$TEXT[(ln)	ln- line number	Returns the image at line number ln in your current editing object (workspace or lookspace). If you do not specify ln, IPF 1100 assumes the value of $C.	$TEXT(30) returns "light blue"
$TRIM (string[,side, character])	string,side - RIGHT, LEFT, BOTH, ALL, BOTH, ALL, character - single character	Removes characters from the specified side (RIGHT, LEFT, BOTH, ALL) If you omit side, IPF 1100 assumes the RIGHT side. If you omit character, IPF 1100 assumes the blank character.	$TRIM("A B C",all) returns "ABC"
$UPPERCASE (string-expression	string-exp	Converts a string-expression to uppercase characters.	$UPPERCASE(cat) returns "CAT"

Section 6
Redirecting Input and Output

This section explains how to have IPF 1100 read your input from a file, instead of waiting for you to enter commands at your terminal. It also shows how IPF 1100 can send to a file the output it normally displays on your screen.

6.1. Standard I/O Paths

Every time you use IPF 1100, three standard I/O paths exist:

- One for IPF 1100 input
- One for IPF 1100 output or all output (including error output, processor output, and IPF 1100 output)
- One for IPF 1100 error output

Normally, IPF 1100 reads input from your terminal and displays normal output and error output on your terminal (unless your job was a batch job, in which case input would be from the card reader, with normal and error output to the high-speed printer).

The IN, OUT, and ERROR commands redirect the standard I/O path (until the end of your IPF 1100 session or subsequent IN, OUT, or ERROR commands).

6.2. Providing Input from a File

You can save a sequence of IPF 1100 commands in a file. For example, you could save a common set of commands you use often. When you want to use these commands, the IN command tells IPF 1100 to read input from this file, instead of from your terminal. The format is

```
IN FILE=filename
```

IPF 1100 subsequently reads input from *filename*. Within a file, you can have the command IN FILE=TERMINAL to tell IPF 1100 to resume reading input from your terminal.

You can also enter

```
IN FILE=filename  ECHO=YES
```

The ECHO=YES keyword parameter tells IPF 1100 to send to your terminal screen every image it reads as input, letting you see a copy of the input data file. You may want to do this when debugging programs. IPF 1100 ignores ECHO=YES when FILE=TERMINAL.

When IPF 1100 reaches the end of *filename*, it reverts the standard input file to its previous assignment.

Here is an example using IN commands:

```
▸ ~C~▸$DISPLAY:=NUMBER      ①
▸ ~C~▸OLD X      ②
▸ ~C~▸PRINT ALL      ③
▸0010.00 DISPLAY "This is the first line."      ④
▸0020.00 DISPLAY "This is the second line."
▸ ~C~▸IN X      ⑤
▸This is the first line.      ⑥
▸This is the second line.
▸ ~C~▸
```

Explanation

① Set the value of the system variable $DISPLAY to NUMBER (that is, you want IPF 1100 to display line numbers).

② Copy into your workspace the file X from your work directory.

③ Print the contents of your workspace.

④ Your workspace consists of two DISPLAY commands.

⑤ Enter an IN command so IPF 1100 will read images from the file X (instead of from your terminal).

⑥ Since file X contains the two DISPLAY commands, IPF 1100 displays the two strings.

Note: *You can only use the IN command for input consisting of IPF 1100 commands.*

For more information on the IN command, see 7.2.15.

6.3. Marking the End of Input

You may have the following scenario:

- You tell IPF 1100 to read input from a file,
- You have an IN FILE=TERMINAL command saved in that file telling IPF 1100 to resume reading input from your terminal, and
- You have more images left in that file (following the IN FILE=TERMINAL command) you want to continue reading.

To continue reading those images, enter

```
END_INPUT
```

For example, suppose you entered an IN FILE=ABC command (see 6.2). Then within file ABC's saved input, there is an IN FILE=TERMINAL command (that is, a nested IN command). To return control back to the input file, you need an END_INPUT command.

```
▶ ~C~▶IN FILE=ABC  ECHO=YES      ①
▶CREATE ACCTS.     ②
▶NEW ACCTS.EAST
▶100 ACCOUNTS
▶200 SMITH 831054
▶300 JONES 772106
▶SAVE
▶IN FILE=TERMINAL
▶ ~C~▶400 LARSON 218805      ③
▶ ~C~▶REPLACE
▶ ~C~▶END_INPUT
▶OLD HIST.WEST     ④
▶ ~C~▶
```

Explanation

① Enter an IN command, redirecting IPF 1100 to read commands from a file (ABC). IPF 1100 echoes the input to your screen because you entered ECHO=YES.

② IPF 1100 reads the next seven lines of input from file ABC. On the first line, a CREATE command tells IPF 1100 to catalog a new file named ACCTS. Next, a NEW command creates a file in your workspace called ACCTS.EAST. The next three lines are data for the program ACCTS.EAST. On the next line, a SAVE command tells IPF 1100 to save this program into a file. Then comes another IN command, this time telling IPF 1100 to temporarily go back to your terminal for more input.

③ IPF 1100 reads one more line of data from your terminal, replaces the file to save the new line of data, and then encounters an END_INPUT command. This tells IPF 1100 to go back to reading input from the file ABC again.

④ IPF 1100 reads an OLD command from file ABC and copies the file HIST.WEST into your workspace. IPF 1100 then resumes reading input from your terminal.

For more information on the END_INPUT command, see 7.2.10.

6.4. Sending Output to a File

Normally, IPF 1100 displays output on your terminal. To save the IPF 1100 output so you can look at it again, enter

```
OUT FILE=filename
```

where *filename* is the name of an existing data file where you want to redirect the IPF 1100 output. Using this form of the OUT command, only the output from IPF 1100 commands is redirected to *filename*.

To save all output (IPF 1100 output, processor output, and all error output) so you may look at it again, enter

```
OUT FILE=filename.  TYPE=ALL
```

where *filename* is the name of an existing data file where you want IPF 1100 to redirect all of the output.

To redirect the output back to your terminal, enter

```
OUT FILE=TERMINAL
```

IPF 1100 can simultaneously display IPF 1100 output images at your terminal, besides sending them to *filename*. (However, you cannot simultaneously display the output images at your terminal when redirecting all output [TYPE=ALL]. IPF 1100 ignores ECHO=YES when FILE=TERMINAL and when TYPE=ALL). To do this, enter

```
OUT FILE=filename.  TYPE=IPF ECHO=YES
```

Notes:

- *The output replaces anything already in filename.*
- *You must specify a data file for filename.*
- *You may not simultaneously display the output at your terminal when redirecting all output (OUT FILE=filename. TYPE=ALL)*
- *You may not redirect all output (OUT FILE=filename. TYPE=ALL) at the same time as you are sending error messages to a file (see 7.2.24).*

Of course, you can read input from one file (by entering an IN command) and send output to another file (by entering an OUT command).

For more information on the OUT command, see 7.2.24.

6.5. Sending Error Messages to a File

To redirect error messages IPF 1100 displays on your terminal, enter

```
ERROR FILE=filename.
```

where *filename* is the name of an existing data file where you want IPF 1100 to direct error output. To redirect error output back to your terminal, enter

```
ERROR FILE=TERMINAL
```

You can also enter

```
ERROR FILE=filename.  ECHO=YES
```

This additional option has the same meaning as described for the OUT command in 7.2.24, except that it pertains to error output.

Notes:

- *The error output replaces anything already in filename.*
- *You must specify a data file for filename.*
- *You may not redirect the IPF 1100 error output to a separate file while an OUT command with TYPE=ALL is in progress.*

Here is an example of using the ERROR and OUT commands.

```
▶ ~C~▶CREATE FILE=OUTFILE.     ①
▶ ~C~▶CREATE FILE=ERRORFILE.
▶ ~C~▶NEW FILE=ABC      ②
▶ ~C~▶10 blue
▶ ~C~▶20 green
▶ ~C~▶OUT FILE=OUTFILE.     ③
▶ ~C~▶ERROR FILE=ERRORFILE.
▶ ~C~▶PRINT ALL      ④
▶ ~C~▶LIST FILE=NOFILE.     ⑤
▶ ~C~▶OUT FILE=TERMINAL     ⑥
▶ ~C~▶ERROR FILE=TERMINAL
▶ ~C~▶OLD FILE=OUTFILE.     ⑦
▶ ~C~▶PRINT ALL      ⑧
▶blue
▶green
▶ ~C~▶OLD FILE=ERRORFILE.      ⑨
▶ ~C~▶PRINT ALL    ⑩
▶**FILE6130 IPF 1100 cannot find the NOFILE. file.
▶ ~C~▶CREATE ALLOUTPUT.
▶ ~C~▶OUT FILE=ALLOUTPUT.  TYPE=ALL    ⑪
▶ ~C~▶LIST FILE=NOFILE.    ⑫
▶ ~C~▶OLD FILE=OUTFILE.
▶ ~C~▶PRINT ALL    ⑬
▶ ~C~▶OUT FILE=TERMINAL    ⑭
▶ ~C~▶OLD FILE=ALLOUTPUT.
▶ ~C~▶PRINT ALL    ⑮
▶**FILE6130 IPF 1100 cannot find the NOFILE. file.
▶blue
▶green
▶ ~C~▶
```

Explanation

① Create two new files, OUTFILE and ERRORFILE.

② Erase and name your workspace ABC, then enter two images into it.

③ Enter OUT and ERROR commands, redirecting IPF 1100 output and error messages to these new files.

④ Enter a PRINT command. Because you redirected output, IPF 1100 does not display anything at your terminal.

⑤ Enter a LIST command, deliberately naming a file that does not exist (NOFILE) so that IPF 1100 displays an error message. Again, IPF 1100 redirects output from your terminal to an alternate file (ERRORFILE).

⑥ Redirect both IPF 1100 output and error messages back to your terminal with another pair of OUT and ERROR commands.

⑦ Retrieve the file OUTFILE using an OLD command.

⑧ Print the contents of your workspace. You can see that it contains the redirected output from your previous PRINT command (at 4).

⑨ Retrieve the file ERRORFILE using another OLD command.

⑩ Print its contents. The error message it contains got there as a result of redirecting error messages earlier, then entering the erroneous LIST command (at ⑤).

⑪ Enter an OUT command to redirect all output, including error messages, to file ALLOUTPUT.

⑫ Enter a LIST command naming a nonexistent file. IPF 1100 will display an error message. The error message will be redirected to the ALLOUTPUT file.

⑬ Enter a PRINT command. IPF 1100 will redirect this output to the ALLOUTPUT file.

⑭ Redirect all output back to your terminal.

⑮ Retrieve the file ALLOUTPUT. using the OLD command and print its content. The error message it contains got there as a result of entering the erroneous LIST command at ⑫, and the other images in the file got there as a result of the PRINT command at ⑬.

Section 7
Command Reference

This section consolidates and expands information about IPF 1100 commands introduced in their most basic format in Sections 2 through 6. It shows all keyword parameters for each command and whether positional notation is allowed.

7.1. Format Conventions

As stated in Section 1, IPF 1100 commands consist of an action word followed by a series of keyword parameters. Refer to Section 1 for detailed notation and format conventions.

IPF 1100 allows a standard abbreviation for all command names, keywords, and predefined user-supplied values except the LOGOFF command, which you cannot abbreviate. The standard abbreviation consists of the first four characters of the command name or the first four characters of each word in a command separated by an underscores. For example, CHANGE_ATTRIBUTE can be abbreviated CHAN_ATTR. Some commands allow 3-character abbreviations. These abbreviations are identified with the command.

Keyword descriptions indicate whether that keyword can be specified using positional notation.

7.2. Command Formats

The following subsections present the IPF 1100 Command Language commands in alphabetical order. The purpose, full format, command description, and examples are given for each command.

7.2.1. ACCEPT Command

The ACCEPT command solicits a value for an IPF 1100 variable. You can abbreviate this command ACC.

Format

```
ACCEPT  VARIABLE=var
        [ PROMPT=string-exp ]
        [ FROM= { USER | OPERATOR } ]
```

where:

var	Specifies a variable (user '%' or system '$') where IPF 1100 should place the user response. This is the first keyword parameter you can specify by position.
string-exp	Specifies what IPF 1100 displays as the prompt. If you do not specify PROMPT, IPF 1100 displays a null string. This is the second keyword parameter you can specify by position.
FROM=	Specifies where the PROMPT (if any) appears. The value USER tells IPF 1100 to display the PROMPT to the user terminal to solicit input. The value OPERATOR directs the PROMPT to the system console, to solicit information from the console operator. You cannot specify this keyword parameter by position.

Description

IPF 1100 determines, based upon the FROM= *keyword parameter*, where to display the PROMPT (if specified). IPF 1100 requests a value (from that source) with which to establish the variable.

Examples

```
▸ ~C~▸ACCEPT VARIABLE=%user1
▸▸2
▸ ~C~▸DISPLAY "VALUE OF %user1= " & %user1
▸VALUE OF %user1= 2
▸ ~C~▸ACCEPT VARIABLE=%u3 PROMPT="ENTER YOUR NAME"
▸ENTER YOUR NAME▸ROSCOE
▸ ~C~▸ACCEPT %SYSTEM, "WHAT SYSTEM IS THIS?" FROM=OPERATOR
▸ ~C~▸DISPLAY "SYSTEM IS: " & %SYSTEM
▸SYSTEM IS: 52A
▸ ~C~▸▒
```

7.2.2. ATTACH Command

The ATTACH command assigns a cataloged file to your session. You can abbreviate this command ATT.

Format

```
ATTACH  FILE=filename
        [ ACTION= ( NONE | WAIT | EXCLUSIVE | PURGE | RECOVER |
                  EXISTENCE ) ]
```

where:

filename	Is the name of the file to attach to your session. This is the first positional keyword parameter.
ACTION=	Indicates any special action you want IPF 1100 to take when attaching the specified file. You may specify a list of choices for this keyword parameter. This is the second positional keyword parameter.
NONE	The default value, indicates that no special action should be taken when attaching the file. IPF 1100 will attach the file if it is available and the file will remain accessible to other users. Do not specify NONE in combination with the other choices of the ACTION keyword parameter list, because it is then meaningless, since it will be overridden by the other specified choices.
WAIT	Indicates that you want to wait until the file is available and can be attached to your session.
EXCLUSIVE	Indicates that you want the file attached exclusively to your session. If you specify this choice, no one else will be able to access the specified file until it is freed from your session.
PURGE	Indicates that you want the file purged from the Master File Directory if the session terminates normally or if the file is freed before the session terminates.
RECOVER	Indicates that the file is to be attached even if it has been disabled.

EXISTENCE	Indicates that you want the file attached only for the purpose of determining if it already exists. The file will be attached regardless of whether it is exclusively attached to another session, disabled, or stored on tape for long term storage. The file will be attached in a read and write inhibited condition. Therefore, you will have to free the file before attempting to read from or update it in any way.

Description

With the ATTACH command, you can explicitly attach a file. This is not normally required (except for cataloged tape files) since IPF 1100 assigns and frees files as needed. You may specify any combination of the ACTION keyword parameter values to have your file attached in a special way. However, you cannot specify the ACTION keyword parameter when attaching an IPF 1100 file within a directory, because the special action applies to your entire directory.

Examples

```
ATTACH FILE=DIREC1.
ATTACH FILE=DATAFILE3. ACTION=EXCLUSIVE
ATTACH FILE1O.,(EXCLUSIVE,WAIT)
ATTACH DIRECTORY1.,WAIT
ATTACH DIREC4.  ACTION=(EXCLUSIVE,PURGE)
```

7.2.3. "Call" Command

The "Call" command calls an IPF 1100 procedure, calls a processor or language compiler (using standard options), or executes an absolute or object module.

Format

```
filename [ arg-set ]
         [ OPTIONS=opt ]
         [ END= { NO | YES } ]
```

where:

filename	Is the name of the file containing the procedure, processor, absolute, or object module you want to call.
arg-set	Is the set of arguments, if any, required by the processor or procedure. You can specify *arg-set* by position.
opt	Is a list of options for *filename*. You cannot specify this keyword parameter by position.
END=	Specifies whether the processor or compiler is to begin executing immediately. END=YES means the processor should begin, and terminate when it is through, so that another IPF 1100 command can be entered. END=NO means that you will be able to enter information to the processor, if it needs it, and also that you may need to enter a special command to cause it to terminate. You cannot specify this keyword parameter by position. The default is NO.

Description

The "Call" command causes IPF 1100 to call the processor in the specified file. If there is no absolute or object module with that name, then IPF 1100 executes the IPF 1100 procedure by that name, if there is one. IPF 1100 saves the *arg-set*, which is obtainable by the executing absolute, object module, or procedure, if the processor or procedure was written to handle and process *arg-set*. The parameter *arg-set* can be a list of up to 20 strings, which you can specify by position or by the keywords F1=, F2=,... F20=. For example, you could compile an ASCII FORTRAN program A.SYM and produce a relocatable A.REL by entering:

```
FTN F1=A.SYM F2=A.REL END=YES
```

To tell ASCII FORTRAN you want a long listing, use the keyword OPTIONS= and the value "L":

```
FTN F1=A.SYM F2=A.REL OPTIONS=L END=YES
```

You will see that calling a procedure, calling a processor, or entering an IPF 1100 command all have the same basic format--just entering their name. If you specify a filename but no directory name, IPF 1100 will search your system, home, and work directories in the order specified by $CALLORDER. (see 4.4.)

Examples

```
ACME*RUN.TOTAL F1=SUB1.  F2=SUB2.  F3=SAVEIT.
ACOB F1=INP F2=OUTP OPTIONS=S END=YES
```

Note: *IPF 1100 allows all files that are read- and write-inhibited to pass through the file assignment checking of the "Call" command. This ensures that Execute-only files can be processed by "Call."*

7.2.4. CONDITION Command

The CONDITION command changes the value of the Run Condition Word.

Format

```
CONDITION   AREA= { T2 | T3 | S3 | S4 | S5 | S6 }
            VALUE=numeric-expression
            [ FUNCTION= { REPLACE | LAND | LOR | LXOR } ]
```

where:

AREA=	Specifies the part of the Run Condition Word that is to be changed. This is the first keyword parameter you can specify by position. You must provide this area.
numeric-expression	Specifies the integer that is to be used to alter the Run Condition Word. This is the second keyword parameter you can specify by position. You must provide this integer.
FUNCTION=	Specifies the logical operation that is to be performed between the portion of the Run Condition Word specified by AREA and *numeric-expression*, before storing the result back into the portion of the Run Condition Word specified by the keyword parameter AREA=. REPLACE will cause the *numeric-expression* to be stored directly. The other functions operate as described in 5.7.4. You can specify the keyword parameter by position.

Examples

```
CONDITION AREA=S6  VALUE=0  FUNCTION=REPLACE
CONDITION T3,8,LOR
```

7.2.5. COPY Command

The COPY command copies data from one file to another. You can abbreviate this command COP.

Format

```
COPY  FROM=filename-1
      TO=filename-2
      [ TYPE= ( ALL | SYMBOLIC | RELOCATABLE | ABSOLUTE | OMNIBUS ) ]
      [ POSITION= { REPLACE | ADD } ]
      [ COUNT= { 1 | exp } ]
      [ RETAIN= { YES | NO } ]
```

where:

filename-1 — Specifies the name of the file IPF 1100 copies the data from. You must provide this name. This is the first keyword parameter you can specify by position.

filename-2 — Specifies the name of the file IPF 1100 copies to. You must provide this name. This is the second keyword parameter you can specify by position.

TYPE= — Specifies the type of file to copy. Specify these values (SYMBOLIC, RELOCATABLE, ABSOLUTE, OMNIBUS, and ALL) when you want to copy specific types of files. The value for this keyword parameter can be a list of file types (for example, TYPE=(SYMBOLIC,RELOCATABLE)). You can specify any combination of the values SYMBOLIC, RELOCATABLE, ABSOLUTE, OMNIBUS, and ALL. The default is ALL. A directory can include as many as four files that have the same name but different types. For copies involving data files, you can only specify ALL or SYMB. This is the third keyword parameter you can specify by position.

POSITION=	Specifies where and how IPF 1100 should store the transferred data in the destination file. That is, the transferred data may be appended to or overwrite the destination file. The value REPLACE means that the transferred data should overwrite the destination file. The value ADD means IPF 1100 should append the transferred data to the destination file. The default value for this keyword parameter is REPLACE. You cannot specify this keyword parameter by position.
COUNT=	For tape-to-tape copies, *exp* is the number of files you want copied. Its value must be a positive integer expression and its default value is one. For print-directed copies, *exp* specifies the number of copies of *filename-1* to send to the print device specified by *filename-2*. Again, its value must be a positive integer expression and its default is one. The maximum value in this second case is 100. You cannot specify this keyword parameter by position.
RETAIN=	Has meaning only for print-directed copies. RETAIN= determines the life of *filename-1* after the copy operation (in this case, actually a print operation) is complete. For the default value of YES, IPF 1100 retains *filename-1* on the system after the print operation is complete. For a value of NO, IPF 1100 removes *filename-1* from the system after the print operation is complete. You cannot specify this keyword parameter by position.

Description

Use the COPY command to copy files and tapes, and send output to a printer. If you specify keyword parameters that are irrelevant for a particular operation, IPF 1100 displays error messages.

Use the keyword parameter POSITION= to accomplish specific results. In most cases, you should specify POSITION=REPLACE. Exceptions are when you are copying one directory to another and want to save the current files of the destination directory, or when you are copying a file to a directory and want to save the current files of the destination directory.

You can use the COPY command with mass-storage files, tape files, and print device files.

- Mass-storage files

  You can copy entire directories, individual files included in directories, or data files. The specified files must already exist. All parameters other than *filename-1* and *filename-2*are optional.

- Tape files

  For copies involving tapes, the tapes must be mounted. IPF 1100 does not allow the form *directory.file* as a filename for *filename-1* or *filename-2*. Do not specify any of the optional keyword parameters of the COPY command, except COUNT=*exp*, for tape-to-tape copies.

- Print device files

  To direct a file to a print device, specify the qualifier DEV$ as part of *filename-2*. The filename appended to DEV$ should be the name of the print device where you want the printing done (for example, COPY FROM=FILE1. TO=DEV$*PR777). The keyword parameter COUNT=*exp*must be an integer from 1 to 100. The keyword parameter RETAIN= specifies the life of *filename-1*. Do not specify any other optional keyword parameters of the COPY command. The *filename-1* can only be a data file or a print file (created by an OUT *file* TYPE=ALL command, or by an Executive BRKPT statement), or a file within a directory.

Examples

```
COPY FROM=SUBTOTAL TO=TOTAL TYPE=SYMBOLIC POSITION=REPLACE
COPY subtotal,total,symbolic position=replace
COPY DIR*A., DIR*B.  POSITION=ADD
COPY FROM=MYFILE. TO=SAVETAPE.
COPY FROM=SAVETAPE.  TO=NEWFILE.
COPY FROM=TAPE1.TO=TAPE2.  COUNT=3
COPY FROM=DATAFILE.  TO=DEV$*PR6.  RETAIN=NO COUNT=10
COPY FROM=DIRFILE.A TO=DEV$*PR1
COPY MYFILE.B,DEV$*PR2 COUNT=5
COPY FROM=ABC TO=DEV$*PR5 RETAIN=NO
```

Here are some examples using the POSITION keyword parameter.

If A and B are both directories and the files of directory A are included with any existing files of directory B, enter the following. (However, any files of directory B with the same name and type as files of directory A are replaced.)

```
COPY FROM=A. TO=B. POSITION=ADD
```

If A and B are both directories and all files of directory B are replaced by the files of directory A, enter

```
COPY FROM=A.  TO=B.  POSITION=REPLACE
```

If A and B are both directories, you want to use the default value of POSITION=REPLACE, and all files of directory B are replaced by the files of directory A, enter

```
COPY FROM=A.  TO=B.
```

If A and B are both data files and you want to copy A to B, overwriting file B with the data in file A, enter

```
COPY FROM=A.  TO=B.  POSITION=REPLACE
```

If A and B are both directories and you want to copy file A.1 to file B.2, enter the following. (If there already is a file B.2, IPF 1100 deletes the old one.)

```
COPY FROM=A.1 TO=B.2 POSITION=REPLACE
```

If A is a directory and B is a data file, and you want to copy file A.1 to file B, enter

```
COPY FROM=A.1 TO=B.  POSITION=REPLACE
```

If A is a directory and B is a data file, and you want to copy file A.1 to file B using the default value of POSITION=REPLACE, enter

```
COPY FROM=A.1 TO=B.
```

If A and B are both directories and you want to include file A.1 with directory B (the other files of directory B remain), enter

```
COPY FROM=A.1 TO=B.  POSITION=ADD
```

If A is a data file and B is a directory, and you want to copy file A. to file B.2

```
COPY FROM=A TO=B.2 POSITION=REPLACE
```

7.2.6. CREATE Command

The CREATE command catalogs a file (with the attributes you specify) on your system. This file is thereafter accessible using those attributes and the filename you specified. You can abbreviate this command CRE.

Format

```
CREATE  FILE=filename
        [ DEVICE_TYPE=d-type ]
        [ VOLUME=volume-list ]
        [ ACCESS= { PRIVATE | PUBLIC } ]
        [ INITIAL_SIZE=exp-1 ]
        [ MAXIMUM_SIZE=exp-2 ]
        [ GRANULARITY_SIZE=exp-3 ]
        [ LIFE= { PERMANENT | TEMPORARY } ]
        [ PARITY= { ODD | EVEN } ]
        [ DENSITY= { 200 | 556 | 800 | 1600 | 6250 } ]
        [ DEVICE_CLASS= { DISK | TAPE } ]
        [ LABEL_FORMAT= { LABELED | UNLABELED | PARTIAL_LABELED } ]
        [ UNITS= { 1 | 2 } ]
        [ TRANSLATE_PROCESSOR= { ASCII | EBCDIC | FLDATA | XS-3 } ]
        [ TRANSLATE_TAPE= { BCD | EBCDIC | FLDATA | XS-3 | ASCII } ]
        [ FORMAT= { A | C } ]
        [ WRITE= { UNSPECIFIED | ENABLE | PROTECT } ]
        [ BLOCK= { OFF | ON | OPTIONAL } ]
        [ COMPRESSION= { OFF | ON | OPTIONAL | EIGHT | NINE | EDRCON |
                         EDRCOPT } ]
```

where:

filename
Specifies the name of the file. This can be a fully qualified filename (see 2.2). If you do not specify *filename*, IPF 1100 rejects the command (no default value exists). This is the first keyword parameter you can specify by position.

d-type
Specifies the device type for the file. This is an explicit class such as F2. The default is your site's system standard default. Actual *d-type* assignment depends on its availability. This is the second keyword parameter you can specify by position.

Note: When creating a disk file, be aware that beginning *d-type*with a "D" (for example, D33) results in a word-addressable file. Beginning *d-type* with any other character, or omitting the keyword parameter, results in a sector-formatted file. This may affect the size of the file as specified by the INITIAL SIZE, MAXIMUM SIZE, and GRANULARITY SIZE keyword parameters.

volume-list	Specifies the name of the volume or volumes on which IPF 1100 should create the file. For disk storage, you could specify a removable disk pack-id. You can submit up to 10 volumes. This is the third keyword parameter you can specify by position.
ACCESS=	This catalogs the file as PUBLIC or PRIVATE. The default is PRIVATE. You cannot specify this keyword parameter by position.
exp-1	Specifies the number of units (increments of *exp-3)* to be initially allocated for this file. If not specified and the storage selected or defaulted to is DISK, then *exp-1* becomes zero. The parameter has no meaning when you select TAPE for the DEVICE_CLASS. You cannot specify this keyword parameter by position.
exp-2	Specifies the maximum number of units (increments of *exp-3*) beyond which the file should not be allowed to expand (that is, before an error occurs). Exceeding this allocation on OS 2200 systems results in notification of that event to you and possible error termination of the activity causing storage limits to be exceeded. You cannot specify this keyword parameter by position.
exp-3	Specifies the size of the units (in 9-bit bytes) used in the storage allocation keyword parameters of this command (such as MAXIMUM_SIZE and INITIAL_SIZE). This does not represent any "physical block" size. The default is 7,168 bytes (one track) for sector-formatted files or four bytes (one word) for word-addressable files. You cannot specify this keyword parameter by position.
LIFE=	Specifies whether you want this file to be TEMPORARY (only exists during the current session) or PERMANENT. The default is PERMANENT. You cannot specify this keyword parameter by position.
PARITY=	Specifies a (recording) parity, ODD or EVEN, to be used in data transfers to and from the specified tape storage media. The default is your site's system standard default. You cannot specify this keyword parameter by position.

DENSITY=	Specifies the density to record the tape file. Tape densities may be 200, 556, 800, 1600, or 6,250 bits per inch. The default is your site's system standard default. You cannot specify this keyword parameter by position.
DEVICE_CLASS=	Specifies a large classification of storage media. If not specified, the default is DISK. You cannot specify this keyword parameter by position.
LABEL_FORMAT=	Specifies whether the tape used for cataloging a file should be labeled or unlabeled. The default is LABELED. The value PARTIAL_LABELED means that you will need to specify the *volume-list* (that is, tape number) the next time you use the tape. The value LABELED means that the next time you use the tape, you must also specify the same *filename*. You cannot specify this keyword parameter by position.
UNITS=	Specifies the number of tape units required. If you do not specify this keyword parameter, IPF 1100 assumes a default of 1. You cannot specify this keyword parameter by position.
TRANSLATE_PROCESSOR=	Specifies the format of data transferred to or from mass storage. You should use the TRANSLATE_PROCESSOR= and TRANSLATE_TAPE= keyword parameters together. The default is your site's system standard default. You cannot specify this keyword parameter by position.
TRANSLATE_TAPE=	Specifies the format of data transferred to or from tape. You should use the TRANSLATE_PROCESSOR= and TRANSLATE_TAPE= keywords together. The default is your site's system standard default. You cannot specify this keyword parameter by position.
FORMAT=	Specifies the data transfer format for the word-to-byte conversion. The two transfer modes are A format (quarter-word mode) C format (8-bit packed mode) The default is your site's system standard default. You cannot specify this keyword parameter by position.

WRITE=	Specifies whether the tape is to be write-protected or write-enabled. The value of UNSPECIFIED means that the operating system will ensure that the tape is write-enabled if the first I/O to the tape at load point is a write function. (The load point is the position near the beginning of the tape where the reading or writing of data begins.) The value ENABLE means that write operations are allowed on the tape. The value PROTECT indicates that write operations on the tape are not allowed. When DEVICE_CLASS=DISK, the only allowable value for WRITE is UNSPECIFIED. The default value is UNSPECIFIED. You cannot specify this keyword parameter by position.
BLOCK=	Controls block numbering, a feature that appends a hardware block number to each data block written to tape. Block numbering is available only for some tape subsystems. The value OFF indicates that blocks should not be numbered. The value ON turns block numbering on. The value OPTIONAL gives you the option of turning on block numbering at a later time. (If you specify BLOCK=OPTIONAL, the system selects a tape drive that supports block numbering, but does not turn that feature on.) This keyword parameter is valid only when DEVICE_CLASS=TAPE. The default value is established by your site. You cannot specify this keyword parameter by position.

COMPRESSION=

Specifies the extent to which tape data will be compressed as it is recorded. This keyword parameter is only valid for some tape subsystems. Depending on the tape subsystem used, the value ON unconditionally enables 9-bit compression or enables 8-bit compression when FORMAT=A, or 9-bit compression when FORMAT=C. The value OFF means no data compression. The value OPTIONAL gives you the option of enabling data compression at a later time. The value EIGHT enables 8-bit data compression. The value NINE enables 9-bit data compression. The value EDRCON enables Enhanced Data Recording Capability (EDRC). The value EDRCOPT gives you the option of enabling EDRC at a later time. (If you specify OPTIONAL or EDRCOPT, the system selects a tape drive that supports compression or EDRC, respectively, but does not turn that feature on.)

This keyword parameter is valid only when DEVICE_CLASS=TAPE. The default value is established by your site. For some device types, compression requires that you enable block numbering by specifying BLOCK=ON. You cannot specify this keyword parameter by position.

Description

Use the CREATE command to catalog a specified disk or tape file. The file then becomes available for subsequent assignment and use in other commands by referring to that filename.

If you specify a volume, IPF 1100 creates the file on that volume. The parameters *exp-1* and *exp-2* specify the storage allocated for the file. The defaults for these parameters are system dependent.

IPF 1100 creates and catalogs the named file according to the specified attributes. It applies default parameters (of the system on which the file is created) as required.

If you specify DEVICE_CLASS=TAPE, IPF 1100 catalogs the named tape file. If you specify both DEVICE_CLASS and DEVICE_TYPE, they must not conflict (for example, one tape, the other disk). If you specify DEVICE_TYPE but not DEVICE_CLASS, the device type overrides the default for DEVICE_CLASS.

The keyword parameters PARITY=, DENSITY=, LABEL_FORMAT=, UNITS=, TRANSLATE_PROCESSOR=, TRANSLATE_TAPE=, FORMAT=, BLOCK=, and COMPRESSION= are for tape files only (that is, do not use them if specifying a disk file).

If you specify DEVICE_CLASS=DISK and the keyword parameter VOLUME=, you must also specify the keyword parameter DEVICE_TYPE.

If you create a temporary file (using the keyword parameter LIFE=TEMPORARY), it exists until you enter a FREE command or end your IPF 1100 and terminal session (using a LOGOFF command and a @FIN statement).

Examples

```
CREATE FILE=HYDRO*OP4.  DEVICE_TYPE=F4 ACCESS=PUBLIC&
INITIAL_SIZE=5000 MAXIMUM_SIZE=20000

crea hydro*op5.,F33 initial_size=6000

CREATE FILE=PLUTO*NOVA.  VOLUME=999A DEVICE_CLASS=TAPE
```

7.2.7. DESCRIBE Command

The DESCRIBE command displays information about the specified directories or data files.

Format

```
DESCRIBE [ FILE=filename-list ]
```

where *filename-list* specifies the names of the directories or data files for which to display information. The default for the FILE keyword parameter is your work directory. This keyword parameter can be specified positionally.

Description

Use the DESCRIBE command to reference information about your directories or data files. IPF 1100 displays the following information:

- External name
- Project identity of your run
- Date file was created
- Account number of your run
- Date of last reference
- Access modes for the file
- Size of storage allocated for the file
- Portion of file storage attached
- Backup information

Examples

```
DESCRIBE FILE=CARS.

DESCRIBE DATA*FILE.

DESCRIBE FILE=(A.,B.,C.,D.)
```

7.2.8. DISPLAY Command

The DISPLAY command displays the values of variables and expressions. You can abbreviate this command DIS.

Format

```
DISPLAY  VALUE=exp
         [ MESSAGE_CLASS= { DATA | REMARK | WARNING  | ERROR
                          | COMMAND | OPERATOR } ]
```

where:

exp	Specifies an expression for IPF 1100 to evaluate and display. This is the first keyword parameter you can specify by position.
MESSAGE_CLASS=	Specifies an output stream where IPF 1100 directs the expression. The value DATA directs the expression to the standard output data stream. The values REMARK, WARNING, and ERROR all direct the expression to the standard error output stream. See 6.4 and 6.5 for information on the output data stream. The value COMMAND directs the expression to your terminal when $FULLSCREEN=FALSE, and to the Command region when $FULLSCREEN=TRUE. The value OPERATOR directs the expression to the system console. The default class is DATA. This is the second keyword parameter you can specify by position.

Description

Use the DISPLAY command to display the value of an expression, a system variable (beginning with $), or a user variable (beginning with %). The MESSAGE_CLASS keyword parameter tells IPF 1100 where to send the expression. When using the DISPLAY command in full-screen mode, remember that control characters are replaced by blanks to prevent destruction of the full-screen display.

Examples

```
DISPLAY VALUE=$USERID
DISPLAY %C
DISPLAY "ENTERING COMPILATION PROCESS..." MESSAGE_CLASS=OPERATOR
```

7.2.9. *END Command

The *END command terminates input mode (automatic line numbering and solicitation of input).

Format

```
*END
```

Description

Use *END to leave INPUT mode in EDIT 1100 (see the *EDIT 1100 User's Guide)*. The * is the default value of the system variable $OMNIPRESENTCHAR. You can change this value, for example, to allow entry of data that requires the characters *END in columns 1-4.

7.2.10. END_INPUT Command

The END_INPUT command indicates the end of input from the alternate file named in the last IN command. You can abbreviate this command END_INP.

Format

```
END_INPUT
```

Description

Use END_INPUT to tell IPF 1100 to stop reading from the current file (for example, your terminal after an IN FILE=TERMINAL command) and resume reading from the previous file. For more information, see 6.3.

7.2.11. ERASE Command

The ERASE command erases a directory or a data file.

Format

```
ERASE FILE=filename-list
```

where *filename-list* is the name of the directory or data file, or a list of directory names or data files, to be erased. This is the only keyword parameter, and you can specify it by position.

Description

Use ERASE to release the contents of a directory or data file; you may then reuse the directory or data file.

The keyword parameter *filename-list* can be a list of directory names or data filenames.

Examples

```
ERASE  FILE = JUNE.
ERASE  COLORS*PRIMARY.
ERASE  (MY*DIRECTORY.,SCRATCH.,CMDL*TCF.)
```

7.2.12. ERROR Command

The ERROR command redirects the standard error output data path. You can abbreviate this command ERR.

Format

```
ERROR FILE=filename
      [ ECHO= { NO | YES } ]
```

where:

filename	Is the name of the file where IPF 1100 should direct the standard error output data path. To redirect this path to your terminal, use TERMINAL. You can specify *filename* by position.
ECHO=	Specifies whether or not (YES or NO) IPF 1100 should simultaneously display at your terminal images directed to the file. The default ECHO=NO, results in the images going only to the specified filename. If you specify FILE=TERMINAL ECHO=NO, IPF 1100 ignores the ECHO keyword (that is, IPF 1100 directs the images to your terminal). You cannot specify this keyword parameter by position.

Description

IPF 1100 normally displays error messages from your IPF 1100 session on your display terminal. You can specify an alternate file for IPF 1100 to send this type of output to using the ERROR command.

Examples

```
ERROR FILE=AMR*SAVE-IT.
ERROR AUDIT-IT.  ECHO=YES
```

7.2.13. EXECUTE Command

The EXECUTE command executes an Executive Control Language (ECL) or IPF 1100 command.

Format

```
EXECUTE COMMAND=string-exp
        [ LANGUAGE= { ECL | IPF } ]
```

where:

COMMAND=	Specifies the command you want to execute, expressed as a string. You cannot specify more than one command. An ECL command must begin with the special character @. This is the first keyword parameter you can specify by position.
LANGUAGE=	Specifies the control language type of the command string you want to execute. LANGUAGE=ECL indicates Executive Control Language and is the default. LANGUAGE=IPF indicates IPF 1100 Command Language. This is the second positional keyword parameter you can specify by position.

Description

The EXECUTE command lets you execute any Executive Control Language command or any IPF 1100 command (except IPF 1100 procedures flow of control commands) by supplying a string expression. You may execute only one ECL or IPF 1100 command at a time. You cannot end the command keyword parameter string with a continuation character. (See 1.9.9 for a complete explanation of string expressions.)

You can use the EXECUTE command to send transparent control statements to the operating system. These statements start with two "at signs" (@@). If you use the EXECUTE command to execute a transparent control statement that produces output in full-screen mode, you may need to refresh the screen when you are finished examining the output. You can refresh your screen by pressing function key 2. For more information on function keys, see the *IPF 1100 EDIT 1100 User's Guide*.

Note: *Because the COMMAND=string-exp cannot be continued, an ECL command is limited to 80 characters.*

Examples

```
EXECUTE COMMAND="@ADD,L MY*RUNSTREAM.ASG-FILES"
EXECUTE "@BRKPT PRINT$/PRINTFILE", ECL
EXECUTE COMMAND="CREATE FILE="& %FNAME LANGUAGE=IPF
EXECUTE "@@CONS D"
EXEC "%A := %B + 1" ,IPF
EXEC "%len := $length(%id "& %i &")", ipf
EXEC "CHAN_ATTR (image1, image2) "&%X,IPF
```

7.2.14. FREE Command

The FREE command frees the assignment of a file or dismounts a volume from a device. You can abbreviate this command FRE.

Format

```
FREE  FILE=filename-list
      [ ACTION= ( NONE | EXCLUSIVE | FILE | NAME | RETAIN_DRIVE ) ]
```

where:

filename-list — Is the name of the directory or data file, or a list of directories or data files, you want to free or dismount. This is the first keyword parameter you can specify by position.

ACTION — Indicates any special action IPF 1100 should take in freeing the specified file. If you specify ACTION=NONE (the default), IPF 1100 takes no special action; it releases the file and any internal names or devices associated with it.

ACTION=EXCLUSIVE releases exclusive use of the file so that other users can access it. The file also remains attached to your run.

ACTION=FILE frees the file but keeps the internal name associated with the file (as specified on a previous USE command).

ACTION=NAME releases only the internal name and keeps the file attached to your run. When ACTION=NAME, the value you specify for FILE is the internal name you associated with the file on a previous USE command.

ACTION=RETAIN_DRIVE frees a tape file but keeps the tape drive assigned to you for further use.

You can specify a list of choices for ACTION. However, the choices NONE and RETAIN_DRIVE may not be used in a list with any other choices. This is the second positional keyword parameter.

Description

IPF 1100 releases the file you specify. (Otherwise, files remain assigned to you until you enter a @FIN statement following a LOGOFF command.) You can use the ACTION keyword parameter to indicate any special action you want IPF 1100 to take when freeing the file.

Examples

```
FREE FILE=QLP*PHONES.
FREE (DAYFILE.,ACCOUNTS.,CLASS*SCHEDULE.)
FREE FILE=KEEP*ME.  ACTION=RETAIN_DRIVE
FREE STATUS*FILE.,EXCLUSIVE
FREE SHORT.,NAME
```

7.2.15. IN Command

The IN command redirects the standard input data path.

Format

```
IN  FILE=filename
    [ ECHO= { NO | YES } ]
```

where:

filename	Specifies the name of the file where IPF 1100 should read input. To redirect IPF 1100 to read input from the terminal, use TERMINAL. You can specify *filename* by position.
ECHO=	Specifies whether or not (YES or NO) IPF 1100 should display on the terminal the images it reads from the input data path. ECHO=YES allows you to see the contents of the input data file (this may be useful when debugging). The default is ECHO=NO. You cannot specify this keyword parameter by position.

Description

By using the IN command, you can add images saved in other files to your job stream. These images must be IPF 1100 commands. When IPF 1100 reaches the end of the specified file, the standard input file reverts to its previous value. For more information, see Section 6.

Example

```
IN FILE=JBR*ARCHIVE1.  ECHO=YES
```

7.2.16. INVENTORY Command

The INVENTORY command displays the names of files associated with your session using the value of SELECT that you specify.

Format

```
INVENTORY SELECT= { ATTACHED | PROJECT_ID }
```

where SELECT= specifies the list of files associated with your session about which you want information. If SELECT=ATTACHED, then the names of all cataloged and temporary files currently attached to your session will be displayed. If SELECT=PROJECT_ID, the names of all files cataloged with the project-id of your session will be displayed. This keyword parameter is required and can be specified positionally.

Description

Use the INVENTORY command to find out which files are associated with your session or which files have been created with the same project-id as that of your session.

7.2.17. LIBRARY Command

The LIBRARY command prepares a directory so that it can be used as a library.

Format

```
LIBRARY [ FILE=directory-name-list ]
```

where *directory-name-list* specifies the name of the directory, or a list of directory names, to be made into a library. The default is the work directory. This is the only keyword parameter, and you can specify it by position.

Description

Use the LIBRARY command to prepare a directory for use as a library, so that other programs can reference the subroutines that exist in the directory. (See Section 4 for more information on executing programs.) The keyword parameter *directory-name-list* can be a list of directory names.

Examples

```
LIBRARY FILE = MY*LIBRARY.
LIBR PERSONNEL.
LIBRARY ( MY*SUBROUTINES.,XYZ*PROGLIB.,EXTERNAL*REFS. )
LIBRARY
```

7.2.18. LIST Command

The LIST command displays information about a directory, or a file included in a directory. You can abbreviate this command LIS.

Format

```
LIST  [ FILE=filename ]
      [ ORDER= { FIRST | LAST } |
      [ COUNT= { ALL | exp ]
      [ TYPE= { ALL | SYMBOLIC | ASMP | COBP | FORP | RELOCATABLE
                | ABSOLUTE | OMNIBUS } ]
      [ SUBTYPE= { ALL | subtype } ]
      [ FORM= { SHORT | LONG } ]
```

where:

filename	Is the name of the file to list. The default value is your work directory. It is the first keyword parameter you can specify by position.
ORDER=	Specifies in what order IPF 1100 should list the files. ORDER=FIRST lists the filenames starting with the first file inserted or replaced into the directory (increasing sequence order). ORDER=LAST lists the files starting with the last file inserted or replaced into the directory (decreasing sequence order). This parameter is optional and the default value is FIRST. It is the second keyword parameter you can specify by position.
COUNT=	Specifies the maximum number of filenames to be listed. Its value must be ALL or a positive integer expression. This parameter is optional. If it is omitted, all the files selected by the values given for TYPE and SUBTYPE will be listed. This is the third keyword parameter you can specify by position.
TYPE=	Tells IPF 1100 you want information on items of this type only. The default is TYPE=ALL. You cannot specify this optional keyword parameter by position.
SUBTYPE=	Tells IPF 1100 you want information on items of this subtype only. The default is SUBTYPE=ALL. You cannot specify this optional keyword parameter by position.

FORM=	Is the amount of detail you want in the directory listing. FORM=SHORT (the default) produces for each file the name, type, and subtype. FORM=LONG produces for each file the name, type, subtype, size, and the last update date and time. This is an optional keyword parameter. You cannot specify this keyword parameter by position.

Description

Use the LIST command to tell you what files are in a directory.

You can control the information displayed by using the keyword parameters. FILE specifies which directory or file to display; ORDER determines the order of the display; COUNT is the maximum number of files to display; TYPE and SUBTYPE restrict which files are to be displayed; and FORM controls the amount of detail displayed.

Note that within a given type (specified by the keyword parameter TYPE), there are a number of subtypes (specified by the keyword parameter SUBTYPE). Valid subtypes include

```
SYM   ELT   ASM   COB   FOR   ALG   MAP   DOC   SEC   SSG   APL   BAS
LSP   PLS   PL1   CTS   FLT   PNC   TCL   MSM   MSD   MAC   APT   PGA
QLP   INL   DCL   SDL   FDP   PSP   PAS   IPF   GSA   MSG   PRT   RPG
ADA   PEER  C     FHLL  LINK  COM   PADS  SSDP  FLDP  ABS   OM    BOM
ZM    SSD
```

The language subtypes that are available on your system are defined by the product SYSLIB. For a complete list of the language subtypes available on your system, refer to the *SYSLIB Programming Reference Manual*for the level of SYSLIB installed on your system.

Acceptable values for absolute subtypes include ABS for a basic mode absolute, OM for an object module, BOM for a bound object module, ZM for a ZOOM, and SSD for a subsystem definition.

Examples

```
LIST FILE=N82*UPPER. TYPE=SYMBOLIC FORM=LONG
LIST PART4
LIST BOOKS.,LAST,15
```

7.2.19. LOGOFF Command

The LOGOFF command terminates your IPF 1100 session. You cannot abbreviate this command.

Format

```
LOGOFF
```

Description

The LOGOFF command terminates only your IPF 1100 session; you remain active on your system until you enter a @FIN statement.

7.2.20. MOUNT Command

The MOUNT command assigns a device to you and mounts a storage volume (either a disk or a tape) on that device.

Format

```
MOUNT  FILE=filename
       VOLUME=volume-list
       DEVICE_CLASS= { DISK | TAPE }
       [ DEVICE_TYPE=d-type ]
       [ PARITY= { ODD | EVEN } ]
       [ DENSITY= { 200 | 556 | 800 | 1600 | 6250 } ]
       [ LABEL_FORMAT= { LABELED | UNLABELED  | PARTIAL_LABELED } ]
       [ UNITS= { 1 | 2 }
       [ TRANSLATE_PROCESSOR= { ASCII | EBCDIC | FLDATA | XS-3 } ]
       [ TRANSLATE_TAPE= { BCD | EBCDIC | FLDATA | XS-3 | ASCII } ]
       [ FORMAT= { A | C } ]
       [ WRITE= { UNSPECIFIED | ENABLE | PROTECT } ]
       [ BLOCK= { OFF | ON | OPTIONAL } ]
       [ COMPRESSION= { OFF | ON | OPTIONAL | EIGHT | NINE | EDRCON |
                        EDRCOPT } ]
```

where:

filename	Specifies the name of the file. If you do not specify *filename*, IPF 1100 rejects this command. This is the first keyword parameter you can specify by position.
volume-list	Specifies the name of the volume where IPF 1100 should mount the file. For disk storage, you could specify a removable pack-id. You can submit up to 10 volumes. This is the second keyword parameter you can specify by position.
DEVICE_CLASS=	Specifies the classification of storage media. This is the third keyword parameter you can specify by position.
d-type	Specifies the device type for the file. This is an explicit class such as F2. The default is your site's system standard default. Actual *d-type* assignment depends on its availability. You cannot specify this keyword parameter by position.
PARITY=	Specifies the recording parity to use in data transfers to and from the specified tape storage media.

	The default is your site's system standard default. You cannot specify this keyword parameter by position.
DENSITY=	Specifies the density at which IPF 1100 should record the data. The default is your site's system standard default. You cannot specify this keyword parameter by position.
LABEL_FORMAT=	Specifies whether IPF 1100 should label the tape. The default is labeled. PARTIAL_LABELED means you must specify *volume-list* (that is, the tape number) the next time you use the tape. LABELED means you must also specify the same *filename* the next time you use the tape. You cannot specify this keyword parameter by position.
UNITS=	Specifies the number of tape units needed. If you omit this keyword parameter, IPF 1100 assumes a default of 1. You cannot specify this keyword parameter by position.
TRANSLATE_PROCESSOR=	Specifies the format of data transferred to or from mass storage. You should specify TRANSLATE_PROCESSOR= and TRANSLATE_TAPE= together. The default is your site's system standard default. You cannot specify this keyword parameter by position.
TRANSLATE_TAPE=	Specifies the format of data transferred to or from tape. You should specify TRANSLATE_PROCESSOR= and TRANSLATE_TAPE= together. The default is your site's system standard default. You cannot specify this keyword parameter by position.
FORMAT=	Specifies the data transfer format for the word-to-byte conversion. The two transfer modes are A format (quarter-word mode) C format (8-bit packed mode) The default is your site's system standard default. You cannot specify this keyword parameter by position.

WRITE=	Specifies whether the tape is to be write-protected or write-enabled. The value of UNSPECIFIED means that the operating system will ensure that the tape is write-enabled if the first I/O to the tape at load point is a write function. (The load point is the position near the beginning of the tape where the reading or writing of data begins.) The value ENABLE means that write operations are allowed on the tape. The value PROTECT indicates that write operations on the tape are not allowed. When DEVICE_CLASS=DISK, the only allowable value for WRITE is UNSPECIFIED. The default value is UNSPECIFIED. You cannot specify this keyword parameter by position.
BLOCK=	Controls block numbering, a feature that appends a hardware block number to each data block written to tape. Block numbering is available only for some tape subsystems. The value OFF indicates that blocks should not be numbered. The value ON turns block numbering on. The value OPTIONAL gives you the option of turning on block numbering at a later time. (If you specify BLOCK=OPTIONAL, the system selects a tape drive that supports block numbering, but does not turn that feature on.) This keyword parameter is valid only when DEVICE_CLASS=TAPE. The default value is established by your site. You cannot specify this keyword parameter by position.
COMPRESSION=	Specifies the extent to which tape data will be compressed as it is recorded. This keyword parameter is only valid for some tape subsystems. Depending on the tape subsystem used, the value ON unconditionally enables 9-bit compression or enables 8-bit compression when FORMAT=A, or 9-bit compression when FORMAT=C. The value OFF means no data compression. The value OPTIONAL gives you the option of enabling data compression at a later time. The value EIGHT enables 8-bit data compression. The value NINE enables 9-bit data compression. The value EDRCON enables Enhanced Data Recording Capability (ERDC). The value EDRCOPT gives you the option of enabling EDRC at a later time. (If you specify OPTIONAL or EDRCOPT, the system selects a tape drive that supports compression or EDRC, respectively, but does not turn that feature on.)

This keyword parameter is valid only when DEVICE_CLASS=TAPE. The default value is established by your site. For some device types, compression requires that you enable block numbering by specifying BLOCK=ON. You cannot specify this keyword parameter by position.

Description

Use the MOUNT command to temporarily assign a specified disk or tape file. The file then is available for use in other commands by specifying its name.

IPF 1100 assigns a temporary file according to the attributes you specify and uses default values for those you do not. If you specify both DEVICE_CLASS and DEVICE_TYPE, they must not conflict (for example, one tape, the other disk). The keyword parameters PARITY=, DENSITY=, LABEL_FORMAT=, UNITS=, TRANSLATE_PROCESSOR=, TRANSLATE_TAPE=, FORMAT=, BLOCK=, and COMPRESSION= are for tape files only (that is, do not use them if specifying a disk file).

The file exists until you enter a FREE command or end your IPF 1100 and terminal session (using a LOGOFF command and a @FIN statement).

Examples

```
MOUNT FILE=OUR*DISK.  VOLUME=112P DEVICE_CLASS=DISK
MOUNT LION.,102a,disk  WRITE=PROTECT
MOUN file=my*tape.  volu=10a devi_clas=tape devi_type=u9v
MOUNT BILLY*GOAT.,99a,tape DEVICE_TYPE=u9v PARITY=Even UNITS=2
MOUNT MY*DISK. VOLU=4364A  BLOCK=ON  COMPRESSION=ON
```

7.2.21. NAME Command

The NAME command changes the name of the workspace. You can abbreviate this command NAM.

Format

```
NAME NEW_NAME=filename
```

where *filename* is the new name you want the workspace to be known as. You can specify *filename* by position.

Description

IPF 1100 changes the workspace name to *filename*. The contents of the workspace remain unchanged. For more information, see Section 3.

Examples

```
NAME NEW_NAME=ZR*FRIDAY-SAVE.
NAME PART2
```

7.2.22. NEW Command

The NEW command erases the contents of the workspace and gives it a new name.

Format

```
NEW FILE=filename
```

where *filename* is the new name of the workspace. You can specify *filename* by position.

Description

IPF 1100 names the workspace *filename*, but leaves the language subtype of the workspace unchanged. For more information, see Section 3.

Examples

```
NEW FILE=SHIP83
NEW SHIP84
```

7.2.23. OLD Command

The OLD command discards the contents of the workspace or the lookspace and replaces it with the contents of a file you specify.

Format

```
OLD FILE=filename
    [ OBJECT= { WORKSPACE | LOOKSPACE } ]
    [ CCS= { CURRENT | FILE | CCS-identifier } ]
```

where:

filename — Is the name of the file whose contents IPF 1100 will copy into the workspace or lookspace. This is the first keyword parameter you can specify by position.

OBJECT= — Specifies where to copy the file. The workspace is used to update the contents of an existing file, or to create and enter images into a new file. The lookspace is independent of the workspace. It allows IPF 1100 to access file images more quickly. The lookspace is used to browse through files. You cannot update files using the lookspace. The default value is the current value of $SWITCH. This is the second keyword parameter you can specify by position.

The current contents of the workspace or lookspace is not affected when you are editing within the other. For more information on using the workspace and lookspace concurrently, see the SWITCH command (7.2.33).

CCS= — Specifies the coded character set (CCS) in which the working area data is to be coded once the OLD command has completed. CCS=CURRENT tells IPF 1100 to use the current value of the system variable $CCS. CCS=FILE tells IPF 1100 to use the CCS of the first significant data image in the file. CCS=CCS-*identifier* allows you to specify a CCS of your own choice, either by name or by its integer identifier, as long as it is recognized on your system.

If the CCS of a file image differs from that of the workspace or lookspace, the OLD command transliterates the image before placing it in the workspace or lookspace. If you enter an OLD command with the system variable $CCS set to one CCS, and the working area ends up in a different CCS, the value of $CCS is changed to match that of the working area.

The CCS keyword parameter applies only when the I18N feature is active for IPF 1100 on your system. Otherwise it is ignored.

Description

IPF 1100 erases the current contents of the workspace or lookspace and copies *filename* into it. After copying the specified file into your workspace or lookspace, IPF 1100 changes the current name and language type to match the file. In addition, the value of the system variable $SWITCH is set to the value of the OBJECT= keyword parameter. If the *filename* does not exist or is not available, IPF 1100 terminates the OLD command and sends you an error message.

Examples

```
OLD FILE=SUBTOTAL OBJECT=WORKSPACE
OLD anyfile OBJECT=LOOK
OLD qual*file.
OLD outfile.,work
```

7.2.24. OUT Command

The OUT command redirects the standard output data path.

Format

```
OUT   FILE=filename
      [ TYPE= { IPF | ALL } ]
      [ ECHO= { NO | YES } ]
```

where:

filename	Is the name of the existing data file where you want IPF 1100 to redirect the output data images. To redirect the output back to the terminal, use FILE=TERMINAL. This is the first positional keyword parameter.
TYPE=	Indicates the type of output data images that IPF 1100 should send to the specified data file. TYPE=IPF indicates that only output from IPF 1100 commands should be sent to the specified data file. TYPE=ALL indicates that all output should be sent to the specified data file. This output includes all IPF 1100 session output including error output, output from interaction with OS 2200 system processors, and user program output. In order to send all output to a specified data file, an ERROR command cannot be in use. The default is TYPE=IPF. This is the second positional keyword parameter.
ECHO=	Specifies whether or not IPF 1100 should simultaneously display at the terminal the images going to the specified file. You cannot use the ECHO keyword parameter when TYPE=ALL; it will be ignored for this case. IPF 1100 also ignores this keyword parameter if FILE=TERMINAL. Otherwise, the default is ECHO=NO. You cannot specify this keyword parameter positionally.

Description

The OUT command allows you to redirect standard output to a file that you specify. When you specify TYPE=ALL, all output (including error output) is redirected. You cannot redirect all output (TYPE=ALL) when an ERROR command is in progress or when you are in full-screen mode. You must first close the error file by entering ERROR TERMINAL or exit full-screen mode. Also, you may only use the ECHO keyword parameter when redirecting IPF 1100 command output, not all output. For more information about the OUT command, see Section 6.

Examples

```
OUT FILE=DATA*ALT.
OUT FILE=ALL*OUTPUT.  TYPE=ALL
OUT IPFOUTPUT.  ECHO=YES
OUT ALLDATA.,ALL
```

7.2.25. PACK Command

The PACK command removes all purged files from a given directory to make more room for new files in that directory.

Format

```
PACK [ FILE=directory-name-list ]
```

where *directory-name-list* specifies the name of the directory, or a list of directory names to be packed. The default is the work directory. This is the only keyword parameter, and you can specify it by position.

Description

Use the PACK command when one of your directories is nearing its maximum size, to make room for the creation of new files in that directory and to allow IPF 1100 to efficiently handle the files in the directory.

The value *directory-name-list* can be a list of directory names.

Examples

```
PACK FILE = LARGE*DIRECTORY.
PACK MY*LIBRARY.
PACK (XYZ*PROGRAMS.,CLASS*ROSTER.,WORKFILE.)
PACK
```

7.2.26. POSITION Command

The POSITION command moves through a tape file.

Format

```
POSITION FILE=filename
         [ COUNT= { 0 | exp } ]
         [ DIRECTION= { FORWARD | BACKWARD } ]
```

where:

filename	Is the name of the tape file you want IPF 1100 to position. The file must already be assigned to your session. This is the first keyword parameter you can specify by position.
exp	Specifies how many files from the tape's current position you want to move. A number greater than zero causes IPF 1100 to move the tape forward that number of files. You can go backward by using the keyword parameter DIRECTION=BACKWARD. The default value for *exp* is zero. This causes IPF 1100 to rewind the tape to its beginning. This is the second keyword parameter you can specify by position.
DIRECTION=	Specifies whether you want IPF 1100 to move the tape forward or backward from its current position. The default is FORWARD. IPF 1100 ignores this keyword parameter if COUNT=0 (or if you do not specify COUNT). This is the third keyword parameter you can specify by position.

Description

The POSITION command lets you move a tape file forward, backward, or rewind it to its beginning. For more information, see 2.20.

Note: *Using this command with multireel tape files is restricted when you specify backward positioning. You cannot move backward to a file beyond the beginning (or load point) of the reel you are currently using. A request to rewind the tape positions you at the beginning of the first reel of a multireel tape file. A request to move forward continues to succeeding reels of a multireel tape file.*

Examples

```
POSITION FILE=MYTAPE.  COUNT=3
POSITION FILE=TAPE6A.  COUNT=0
POSITION TAPE6A., 0
posi t3422.,1,backward
```

7.2.27. PURGE Command

The PURGE command deletes data files, directories, and the symbolic, relocatable, absolute, and omnibus files within a directory. This provides a quick method of cleansing your system of unneeded files. You can abbreviate this command PUR.

Format

```
PURGE FILE=filename
      [ TYPE= ( SYMBOLIC | RELOCATABLE | ABSOLUTE | OMNIBUS | ALL ) ]
```

where:

filename	Specifies the name of the file you want to delete. If you do not specify *filename*, IPF 1100 rejects this command. This is the first keyword parameter you can specify by position.
TYPE=	Specifies the type or types of files to delete. You can use this optional keyword parameter only when deleting a file within a directory. You may specify any combination of the four file types or ALL. The default is TYPE=SYMBOLIC. This is the second keyword parameter you can specify by position.

Description

You must supply the name of the file you want deleted, and it must meet the access and security requirements of the system. The TYPE keyword parameter is optional. IPF 1100 deletes the specified file type of the specified file and rejects subsequent references to that specific file.

IPF 1100 provides a level of protection to ensure that unauthorized deletion of files does not occur. Incompatibilities between command specification and file definition result in IPF 1100 rejecting the command and displaying an appropriate message.

You can specify only one filename on each PURGE command. Purging of files follows the rules of the system on which you created or cataloged the file.

Examples

```
PURGE FILE=CAT*DIREC.
PURGE FILE=CAT*DIREC2.SYMFILE TYPE=SYMB
PURGE TGH*DATAFILE.
PURGE DIREC5.FILE1,(SYMB,RELO,ABSO)
```

7.2.28. REPLACE Command

The REPLACE command copies the workspace into an existing file. You can abbreviate this command REP.

Format

```
REPLACE [ FILE=filename ]
```

where filename is the name of the file where IPF 1100 should save the workspace. You can specify *filename* by position.

Description

The REPLACE command operates in the same way as the SAVE command, but is for updating an existing file. You cannot use the REPLACE command on your lookspace.

If you do not specify *filename*, IPF 1100 uses the current name of the workspace. The workspace name does not change after a REPLACE command. For additional information, see 3.6.

Examples

```
REPLACE FILE=RST*ADMIN.WEEK
REPLACE CALC
REPLACE DATAFILE-2.
```

Example using $WORKDIRECTORY:

```
SET $WORKDIRECTORY:="STAR."
OLD FILE=COLORS
10 RED
20 BLUE
30 YELLOW
SET $WORKDIRECTORY:="PAINT."
```

To replace the file COLORS in the directory specified by $FILENAME (STAR), enter

```
REPLACE
```

To replace the file COLORS in the directory specified by the value of $WORKDIRECTORY (PAINT), enter

```
REPLACE COLORS
```

To replace the file COLORS in the directory called A*DIR, enter

```
REPLACE A*DIR.COLORS
```

7.2.29. REVIEW Command

Use the REVIEW command to retrieve and display a list of all user variables or all system variables.

Format

```
REVIEW VARIABLES= { USER | SYSTEM }
```

where VARIABLES= specifies the list of variables (user or system) to be displayed. Each variable name will be displayed with its value. You can specify this keyword parameter by position.

Description

The REVIEW command displays a requested list of variables (user or system) with their values in the following format:

```
variable-name-1        value
variable-name-2        value
     .
     .
     .
```

If you include the keyword parameter VARIABLES=USER, the resulting display contains all the user variables you have created. If you include the keyword parameter VARIABLES=SYSTEM, the resulting display contains all the system variables that you are allowed to see.

Examples

```
REVIEW VARIABLES=USER
REVI SYSTEM
```

Results:

In the first example, IPF 1100 displays all of the user variables that you created. Each variable appears with its current value.

In the second example, IPF 1100 displays all of the system variables, along with the current value of each variable.

7.2.30. RUN Command

Use the RUN command to compile, collect (or link), and execute a symbolic program either in your workspace or in a directory.

Format

```
RUN  [ FILE=filename ]
     [ COMPILER=compiler-name ]
     [ LIBRARY=library-name-list ]
```

where:

filename	Is the name of the file containing the program you want to execute. If you do not specify filename, IPF 1100 uses your workspace. This is the first keyword parameter you can specify by position.
compiler-name	Is the name of the file (included in a directory) containing the compiler you want used when IPF 1100 compiles your program. This is the second keyword parameter you can specify by position.
library-name-list	Is the name of the file, or a list of filenames, containing the libraries you want used when IPF 1100 collects your program. You can submit up to eight libraries. This is the third keyword parameter you can specify by position.

Description

The RUN command executes a specified program (symbolic) from a directory. If you do not specify the keyword parameter FILE=, IPF 1100 executes the contents of your workspace.

If you do not specify the keyword parameters COMPILER= and LIBRARY=, IPF 1100 uses the default system compiler associated with the language type of the file, as well as the default system library.

The RUN command compiles the symbolic you specify according to its language type, or by using the compiler you specify. If you do not specify a compiler (that is, if you do not use COMPILER=), IPF 1100 looks for the necessary compiler in the system library. IPF 1100 collects the program using the new relocatable and the library specified if no object module was produced. If you do not specify a library, IPF 1100 assumes the system library. Then it executes the program (absolute file or object module). IPF 1100 does not save the relocatable file, absolute file, or object module. For more information, see 4.2.

Notes:

- *IPF 1100 allows all files that are read- and write-inhibited to pass through the file assignment checking of the RUN command. This ensures that execute-only files can be processed by RUN.*
- *For some language types, both UCS and non-UCS compilers are available. If your symbolic program falls into this category and you do not specify a value for the keyword parameter COMPILER, IPF 1100 uses the $UCSCOMPILER system variable to select the appropriate default compiler. If $UCSCOMPILER is set to TRUE, the UCS compiler is selected. If $UCSCOMPILER is set to FALSE, the non-UCS compiler is selected.*

Examples

```
RUN FILE=TEST-IT
RUN TEST-IT
RUN FILE=NUMBER.COUNT
RUN FILE=NUMBER.COUNT COMPILER=FTN.FTN LIBRARY=REL*LIBRARY.
RUN NUMBER.COUNT,FTN.FTN,REL*LIBRARY.
```

7.2.31. SAVE Command

The SAVE command copies the workspace into a file. You can abbreviate this command SAV.

Format

```
SAVE [ FILE=filename ]
```

where *filename* is the name of the file where IPF 1100 should save the workspace. You can specify filename by position.

Description

The SAVE command saves the workspace into a file within a directory. (You cannot specify a data file.) If the file within the directory already exists, IPF 1100 does not try to save it, but displays an error message. You cannot use the SAVE command on your lookspace.

If you do not specify *filename*, IPF 1100 uses the current name of the workspace. The workspace name does not change after a SAVE command. For more information, see 3.4.

Examples

```
SAVE FILE=LJR*NEW.DATA
SAVE B-AVG
SAVE
```

Example using $WORKDIRECTORY

```
NEW FILE = STAR.COLORS
10 GREEN
20 PURPLE
30 ORANGE
$WORKDIRECTORY:="PLASTIC".
```

To use the value of $FILENAME (STAR.COLORS) to save a copy of your workspace, enter

```
SAVE
```

To use the value of $WORKDIRECTORY (PLASTIC.) as the directory in which to save the file COLORS, enter

```
SAVE COLORS
```

To save a copy of your workspace in A*DIR.COLORS, enter

```
SAVE A*DIR.COLORS
```

7.2.32. SET Command

The SET command establishes a value for a system or user-defined variable.

Format

```
[ SET ] variable:=exp
```

where:

variable	Is a system variable (beginning with $) or a user variable (beginning with %).
exp	Is an expression.

You can omit typing SET.

Description

IPF 1100 evaluates *exp* and stores the result in *variable*. For more information, see Section 5.

Examples

```
SET $DISPLAY:=NUMBER
SET %N:=4
$DISPLAY:=NONUMBER
```

7.2.33. SWITCH Command

The SWITCH command changes the current editing object to either the workspace or the lookspace.

Format

```
SWITCH OBJECT= { WORKSPACE | LOOKSPACE }
```

where OBJECT= specifies the new current editing object, either the workspace or the lookspace. The workspace is used to update the contents of an existing file, or to create and enter images into a new file.

The lookspace is independent of the workspace. It allows IPF 1100 to access files quickly. The lookspace is used to browse through files. You cannot update files using the lookspace.

The current contents of the workspace or lookspace is not affected when editing within the other. You can specify this keyword parameter by position.

Description

IPF 1100 maintains two editing objects, the workspace and the lookspace. You may switch between the two as often as you wish within your IPF 1100 session. For example, if you are currently editing your workspace, entering SWITCH LOOKSPACE will change the editing object to your lookspace. Your lookspace must have a file in it before IPF 1100 will accept the SWITCH command. To copy a file into the lookspace, use the OLD command.

Examples

```
SWITCH LOOK
SWITCH OBJECT=WORKSPACE
```

7.2.34. TYPE Command

The TYPE command explicitly sets or changes the language subtype of the workspace. You can abbreviate this command TYP.

Format

```
TYPE LANGUAGE=language
```

where LANGUAGE= is the desired language subtype of the workspace. The legal values for this field include all values currently allowed for program file language subtypes as defined by SYSLIB. You can specify this keyword parameter by position. Valid subtypes include

```
SYM   ELT   ASM   COB   FOR   ALG   MAP   DOC   SEC   SSG   APL   BAS
LSP   PLS   PL1   CTS   FLT   PNC   TCL   MSM   MSD   MAC   APT   PGA
QLP   INL   DCL   SDL   FDP   PSP   PAS   IPF   GSA   MSG   PRT   RPG
ADA   PEER  C     FHLL  LINK  COM   PADS  SSDP  FLDP
```

For a complete list of the language subtypes available on your system, refer to the *SYSLIB Programming Reference Manual* for the level of SYSLIB installed on your system.

Description

IPF 1100 updates the language subtype attribute of the workspace so that a subsequent SAVE or REPLACE command updates the actual file's subtype. If you never use a TYPE command or retrieve a file (via an OLD command) that has a subtype other than SYM, the workspace remains subtype SYM.

Examples

```
TYPE LANGUAGE=SYM
TYPE LANG=IPF
TYPE IPF
TYPE FOR
```

7.2.35. USE Command

You can call a data file or directory by any internal name you want during an IPF 1100 session by entering a USE command.

Format

```
USE NAME=identifier
    FILE=filename
```

where:

identifier	Is the name you want associated with the data file or directory. This name can be up to 12 characters long. This is the first keyword parameter you can specify by position.
filename	Is the data file or directory you want to associate the identifier with. (It cannot be a file within a directory). This *filename* must be followed by a period, and optionally preceded by a qualifier. This is the second keyword parameter you can specify by position.

Description

The USE command lets you refer to a data file or directory by another name until you associate another data file or directory by the same identifier or free the file. For more information, see 2.15.

Examples

```
USE NAME=accounts.  FILE=new*entries.
USE NAME=short FILE=long*windedname.
USE short,long*windedname.
```

7.2.36. XQT Command

The XQT command executes a user program (absolute file or object module).

Format

```
XQT FILE=filename
    [ OPTIONS=opt ]
    [ END= { NO | YES } ]
```

where:

filename	Is the name of a file containing the absolute or object module you want to execute. You can specify *filename* by position.
opt	Is the list of options desired for the program you want to execute. You cannot specify this keyword parameter by position.
END=	Specifies whether you will be able to enter input to the program being executed. END=YES means that IPF 1100 tells the program that you will not be entering any input for the program to process. It will begin executing and will terminate when it is done, so you can enter another IPF 1100 command. END=NO (the default) means that you can enter input at the terminal for the program to process, if the program was written to accept such input. You cannot specify this keyword parameter by position.

Description

Use the XQT command to execute application programs you previously used the "Call" command to compile and collect (see 7.2.3).

You can also use the XQT command to execute old absolutes or object modules created in the OS 2200 environment.

Notes:

- *If you specially build your program to handle parameters (make it a processor), you can also use the "Call" command to execute your program.*
- *IPF 1100 allows all files that are read- and write-inhibited to pass through the file assignment checking of the XQT command. This ensures that execute-only files can be processed by XQT.*

Examples

```
XQT FILE=CM.DIPLO END=YES
XQT AMORTIZE
```

Bibliography

OS 2200 Exec System Software Executive Requests Programming Reference Manual (7830 7899). Unisys Corporation.

OS 2200 Interactive Processing Facility (IPF 1100) Administration Guide (7833 3754). Unisys Corporation.

OS 1100 Interactive Processing Facility (IPF 1100) EDIT 1100 User's Guide (7833 3812). Unisys Corporation.

OS 2200 Interactive Processing Facility (IPF 1100) Error Message Reference Manual (7833 4018). Unisys Corporation.

OS 2200 Interactive Processing Facility (IPF 1100) Master Index (7833 3747). Unisys Corporation.

OS 2200 Interactive Processing Facility (IPF 1100) Procedures User's Guide (7833 3770). Unisys Corporation.

OS 1100 Interactive Processing Facility (IPF 1100) User Assistance (UA 1100) User's Guide (7833 3762). Unisys Corporation.

OS 2200 Interactive Processing Facility (IPF 1100) User's Reference Handbook (7833 3820). Unisys Corporation.

OS 2200 Interactive Processing Facility (IPF 1100) User's Quick-Reference Guide (7833 3804). Unisys Corporation.

OS 1100 Meta-Assembler (MASM) Programming Reference Manual(7830 8269). Unisys Corporation.

OS 1100 UDS Relational Data Management System (UDS RDMS 1100) and IPF SQL Interface End Use Guide (7831 0778). Unisys Corporation.

OS 2200 UDS Relational Data Management System (UDS RDMS 2200) SQL Programming Reference Card (7831 0943). Unisys Corporation.

OS 2200 UDS Relational Data Management System (UDS RDMS 2200) SQL Programming Reference Manual (7830 8160). Unisys Corporation.

OS 2200 Distributed Data Processing File and Job Transfer (DDP-FJT) Implementation and Administration Guide (3787 3544). Unisys Corporation.

OS 2200 Distributed Data Processing File and Job Transfer (DDP-FJT) IPF and Batch Interface Operations Guide (3787 3551). Unisys Corporation.

OS 2200 System Services Routines Library (SYSLIB) Programming Reference Manual (3733 1733). Unisys Corporation.

OS 2200 Executive Control Language (ECL) and FURPUR Reference Manual (7830 7949). Unisys Corporation.

Index

A

Abbreviations, 1-11
$ABS function, 5-14
ACCEPT command, 5-9, 7-1
AND logical function, 5-17
ASCII characters
  of an integer, 5-16
  converting to integers, 5-16
$ASCII function, 5-16
Assigning files, 2-17, 7-3
ATTACH command, 2-17, 2-25, 7-3

B

Binary operators, 1-16
Boolean operators, 1-17

C

"Call"command, 4-6, 7-5
Calling IPF 1100, 1-19
Calling processors, 7-5
$CALLORDER variable, 4-8
Cataloging files, 7-12
Cataloging tape files, 2-25
$CHARACTER function, 5-16
OUT command, 6-5, 7-39
IN command, 7-26
Command format, 1-9, 7-1
Command Language
  overview, 1-3
Commands
  ACCEPT, 5-9, 7-1
  ATTACH, 2-17, 2-25, 7-3
  "Call", 4-6, 7-5
  CONDITION, 4-13, 7-7
  COPY, 2-11, 2-24, 7-8
  CREATE, 2-6, 2-25, 7-12
  DESCRIBE, 2-20, 7-18
  DISPLAY, 5-7, 7-19
  *END, 7-20
  END_INPUT, 6-3, 7-20
  ERASE, 2-16, 7-21
  ERROR, 6-6, 7-22
  EXECUTE, 4-12, 7-23
  FREE, 2-19, 2-25, 7-24
  IN, 6-1, 7-26
  INVENTORY, 2-21, 7-27
  LIBRARY, 4-3, 7-27
  LIST, 2-9, 7-28
  LOGOFF, 7-30
  MOUNT, 2-23, 7-31
  NAME, 3-11, 7-35
  NEW, 3-4, 7-36
  OLD, 3-1, 3-8, 7-37
  OUT, 6-5, 7-39
  PACK, 2-15, 7-40
  POSITION, 2-23, 7-41
  PURGE, 2-10, 7-43
  REPLACE, 3-9, 7-44
  REVIEW, 5-8, 7-45
  RUN, 4-3, 7-46
  SAVE, 3-6, 7-48
  SET, 5-7, 7-49
  SWITCH, 3-1, 7-50
  TYPE, 3-12, 7-51
  USE, 2-16, 7-52
  XQT, 4-10, 7-53
Commands, components
  abbreviations, 1-11
  format, 1-9, 7-1
  keyword parameters, 1-8, 1-10
$COMMENTCHAR variable, 1-12
Comment lines, 1-12
Compiling programs, 4-6
Completion notices, 1-5
$COMPLETIONS variable, 1-5
Complex expressions, 5-19
Components, IPF 1100, 1-1
CONDITION command, 4-13, 7-7
$CONDITION function, 4-13
CONDITION function, 5-17

Conditions, 1-13
Constants, 1-13
$CONTCHAR variable, 1-12
Continuation lines, 1-12
COPY command, 2-11, 2-24, 7-8
Copying files, 2-11, 7-8
Copying tape data, 2-24
CREATE command, 2-25, 7-12
Creating directories, 2-6
Creating files, 2-6, 3-4
Cursor positioning, 1-3
Cycles, in files, 2-3

D

Data files, 2-3
Data images, 2-6
Defense Data Network (DDN 1100), 1-2
Deleting files, 2-10, 7-43
$DELIMCHAR variable, 1-14
DESCRIBE command, 2-20, 7-18
Directories
  creating, 2-6
  erasing, 7-21
  home, 2-4
  removing files from, 2-15
  search order of, 4-8
  used as libraries, 4-3, 7-27
  work, 2-5
DISPLAY command, 5-7, 7-19
Displaying file information, 2-20, 7-18
Displaying session filenames, 2-21, 7-27
$DISPLAY variable, 3-5
Distributed Data Processing (DDP 1100), 1-2

E

Editing files, 3-3
END_INPUT command, 6-3, 7-20
*END command, 7-20
Ending input mode, 7-20
ERASE command, 2-16, 7-21
Erasing directories, 2-16, 7-21
Erasing files, 2-16, 7-21
ERROR command, 6-6, 7-22
EXECUTE command, 4-12, 7-23
Executing commands, 4-12, 7-23
Executing programs, 7-53
Expressions
  complex, using, 5-19
  description, 1-14
  full, 1-15
  string, 1-14
  variables in, 1-17

F

feature
  I18N, 5-11
$FIELDSTATUS function, 5-19
$FILE function, 5-18
Filenames, 2-1
Files
  assigning, 2-17
  browsing through, 3-1, 3-14
  cataloging, 7-12
  copying, 2-11, 7-8
  creating, 2-6, 3-4, 7-36
  data, 2-3
  deleting, 2-10, 7-43
  description, 2-1
  displaying information about, 7-18
  displaying information from, 2-20
  displaying names of, 7-27
  editing, 3-1, 3-3
  erasing, 2-16, 7-21
  fully qualified names, 5-18
  in IPF 1100 sessions, 2-21
  listing, 2-9, 7-28
  mass-storage, 7-9
  naming, 2-1
  print device, 7-9
  printing, 2-13
  private, 2-7
  public, 2-7
  read and write keys, 2-2
  reading input from, 6-1, 7-26
  releasing, 2-19, 7-24
  removing, 2-15, 7-40
  retrieving, 3-8, 7-37
  saving, 3-6, 3-9, 7-44
  sending messages to, 6-6, 7-22
  sending output to, 6-5, 7-39
  specifying end of input, 6-3, 7-20
  tape, 7-9
  temporary, 2-8
  transferring to tape, 2-22
  using cycles in, 2-3
  using temporary names, 2-16, 7-52

version names, 2-3
Format conventions
comments, 1-12
conditions, 1-13
constants, 1-13
continuation lines, 1-12
expressions, 1-14
general, 1-12, 7-1
identifiers, 1-13
labels, 1-13
multiple commands, 1-12
strings, 1-14
$FRACTION function, 5-15
FREE command, 2-19, 2-25, 7-24
Full expressions, 1-15
Fully qualified filenames, 5-18

G

Getting field information, 5-19
Getting help, 1-4

H

Home directory, assigning, 2-4

I

Identifiers, 1-13
IN command, 6-1
$INTEGER function, 5-15
Interprogram communication, 4-12
INVENTORY command, 2-21, 7-27
IPF 1100
calling, 1-19
components, 1-1
features, 1-1
I/O paths, 6-1
leaving, 1-19
product description, 1-1
terminating sessions, 7-30
using commands, 1-8
variables, 5-1

K

Keyword notation, 1-18
Keyword parameters
format, 1-10
lists in, 1-17

L

Labels, 1-13
Leaving IPF 1100, 1-19
$LENGTH function, 5-10
LIBRARY command, 4-3, 7-27
LIST command, 2-9, 7-28
Listing files, 2-9, 7-28
Listing IPF 1100 files, 2-21, 7-27
Listing variable names, 5-8, 7-45
Logical functions
AND, 5-17
OR, 5-17
XOR, 5-18
LOGOFF command, 7-30
Lookspace
commands in, 3-14
definition, 3-1, 3-14
restrictions, 3-14
$LOWERCASE function, 5-11

M

Mass-storage files, 7-9
Messages, receiving, 1-5
MOUNT command, 2-23, 7-31
Mounting tapes, 2-23, 7-31
Moving through tapes, 2-23, 7-41
$MULTICMDCHAR variable, 1-12
Multiple commands, 1-12

N

NAME command, 3-11, 7-35
Naming files, 2-1
NEW command, 3-4, 7-36
Notation
keyword, 1-18
positional, 1-18
$NUMBER function, 5-16
Numbers

converting to strings, 5-15
finding absolute values, 5-14
finding fractional portions, 5-15
finding integer portions, 5-15

O

OLD command, 3-1, 3-8, 7-37
$OMNIPRESENTCHAR variable, 4-7
Omnipresent commands, specifying, 4-7
Operators
binary, 1-16
Boolean, 1-17
unary, 1-16
OR logical function, 5-17
Output paging, 1-6

P

PACK command, 2-15, 7-40
$PAD function, 5-13
Page breaks, 1-6
$PAGECALL variable, 1-8
$PAGELENGTH variable, 1-7
$PAGEWIDTH variable, 1-7
$PAGING variable, 1-7
Positional notation, 1-18
POSITION command, 2-23, 7-41
Print device files, 7-9
Printing files, 2-13
Private files, 2-7
Programs
compiling, 4-6
executing, 7-53
file types, 4-2
MASM and Exec, 4-12
running, 4-1, 4-10, 7-46
Public files, 2-7
PURGE command, 2-10, 7-43

R

Read keys, 2-2
Releasing files, 2-19, 7-24
Releasing tape files, 2-25, 7-24
Relocatable libraries, 4-3
Removing old files, 2-15, 7-40
REPLACE command, 3-9, 7-44
Retrieving files, 3-8, 7-37
RETURN key, 1-4
REVIEW command, 5-8, 7-45
RUN command, 4-3, 7-46
Run Condition Word
changing value of, 4-13, 7-7
definition, 4-12
displaying parts of, 5-17
Running programs
methods of, 4-1
steps for, 4-1, 4-11

S

SAVE command, 3-6, 7-48
Saving files, 3-6, 7-44
Screen lines, 2-6
Scrolling the display, 1-3
$SEARCH function, 5-10
SET command, 5-7, 7-49
Signing off your system, 1-20
Specifying a search order, 4-8
Specifying system functions, 5-9
Standard I/O paths, 6-1
String expressions, 1-14
$STRING function, 5-15
Strings
in command format, 1-14
converting to lowercase, 5-11
converting to numbers, 5-16
converting to uppercase, 5-12
finding length of, 5-10
finding line numbers, 5-14
integer representation, 5-16
"padding", 5-13
removing blanks from, 5-12
returning substrings, 5-11
substring position in, 5-10
$SUBSTRING function, 5-11
Subtypes, changing, 3-12, 7-51
SWITCH command, 3-1, 7-50
$SWITCH variable, 3-1
System functions
$ABS, 5-14
$ASCII, 5-16
$CHARACTER, 5-16
$CONDITION, 4-13, 5-17
$FIELDSTATUS, 5-19
$FILE, 5-18
$FRACTION, 5-15

$INTEGER, 5-15
$LENGTH, 5-10
$LOWERCASE, 5-11
$NUMBER, 5-16
$PAD, 5-13
$SEARCH, 5-10
$STRING, 5-15
$SUBSTRING, 5-11
$TEXT, 5-14
$TRIM, 5-12
$UPPERCASE, 5-12
format, 5-9
specifying, 5-9
table of, 5-19
System variables
$CALLORDER, 4-8
$COMMENTCHAR, 1-12
$COMPLETIONS, 1-5
$CONTCHAR, 1-12
$DELIMCHAR, 1-14
$DISPLAY, 3-5
$MULTICMDCHAR, 1-12
$OMNIPRESENTCHAR, 4-7
$PAGECALL, 1-8
$PAGELENGTH, 1-7
$PAGEWIDTH, 1-7
$PAGING, 1-7
$SWITCH, 3-1
$WAITTIME, 1-8
establishing values for, 5-7
listing variable names, 5-8
printing values of, 5-7
table of, 5-1
using, 5-1

T

Tape files, 7-9
Tapes, using
cataloging files, 2-25, 7-12
copying data, 2-24, 7-8
mounting, 2-23, 7-31
moving through, 2-23, 7-41
overview, 2-22
releasing files, 2-25, 7-24
TCP/IP Application Services (TAS), 1-2
Temporary filenames, 2-16, 7-52
Temporary files, 2-8
$TEXT function, 5-14
Transferring files to tape, 2-22, 7-31
$TRIM function, 5-12
TYPE command, 3-12, 7-51
Types
absolute, 4-2
object module, 4-3
relocatable, 4-2
symbolic, 4-2

U

Unary operators, 1-16
$UPPERCASE function, 5-12
USE command, 2-16, 7-52
User variables, assigning, 5-9
Using IPF 1100 commands, 1-8

V

Variables
displaying values, 7-19
establishing values for, 5-7, 7-49
in expressions, 1-17
listing, 5-8, 7-45
soliciting values for, 5-9, 7-1
types of, 5-1
Version names, of files, 2-3

W

$WAITTIME variable, 1-8
Work directory, defining, 2-5
Workspace
assigning, 3-2
bringing files into, 3-8, 7-37
changing name of, 3-11
changing subtype of, 3-12, 7-51
changing the name of, 7-35, 7-51
creating files in, 3-4, 7-36, 7-37
definition, 3-1
saving files from, 3-6, 3-9, 7-48
system variables in, 3-5
Write keys, 2-2

X

XMIT key, 1-3
XOR function, 5-18
XQT command, 4-10, 7-53